UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Biomedical Semantic Question and Answering System

Miguel Jorge Vitorino Sousa Rodrigues

MESTRADO EM INFORMÁTICA

Dissertação orientada por:
Prof. Doutor Francisco José Moreira Couto

2017

Acknowledgments

This is the result of a long, arduous work which would never happen if not with the help, emotional support, and prayers of very special people in my life. This path went throughout eight years of college, beginning with a downward path, followed by a strenuous, uphill but successful battle as each year went by in the five years I spent at FCUL. For those who helped me throughout:

- to every Professor that I had at FCUL along these years, I would not know what I know today in Informatics if not because of your work and teaching.
- to all my colleagues, that were a great source of healthy and refreshing competition year by year, setting the bar higher with each passing year, and to Evandro for his fresh look towards life.
- to my friends and brothers is faith, for being a good source to vent from time to time and for their prayers and for remembering me, I salute, among others, Gui, Pedro, and Jorge.
- to my supervisor in this thesis, Francisco Couto, for his patience, availability, understanding, even though I went AWOL from time to time, and for his readiness to motivate me and give a positive word whenever I feltt down or disappointed.
- to my special and long-time colleague Falé, throughout our countless tasks together, for his preparedness, effort beyond any colleague I have ever worked with, for always being concerned with delivering top results that made me always be on my game, and his bizarre humor since our second year college, a partnership and friendship that still stands.
- to Aline, for being my companion for many years, listening to my agonies, for her support and caring no matter what I went through and no matter at what time I needed her, even from before my adventure at FCUL.
- to my ever supporting family, brother Davide, father Jorge and mother Amélia, for being the exact family that I needed in my life. For being the example of unwavering work, for my second chance, for their love and sacrifices without end, for being

there every step of the way and for being the best example of endurance and biblical love that I have witnessed.

- to the Almighty God, for providing me with all these companions and family, for saving me by His grace alone through faith alone, and for crafting in me a personality that each day is more and more according to His Son, *Soli Deo Gloria*.

After a long thought process of weighting the continuity of my academical journey, I have finally decided to at least halt it as of here, with this thesis.

”Honor your father and your mother,

that your days may be long in the land

that the Lord your God is giving you.”

Exodus 20:12

Resumo

Os sistemas de *Question Answering* são excelentes ferramentas para a obtenção de respostas simples e em vários formatos de uma maneira também simples, sendo de grande utilidade na área de *Information Retrieval*, para responder a perguntas da comunidade online, e também para fins investigativos ou de prospeção de informação. A área da saúde tem beneficiado muito com estes avanços, auxiliados com o progresso da tecnologia e de ferramentas delas provenientes, que podem ser usadas nesta área, resultando na constante informatização destas áreas.

Estes sistemas têm um grande potencial, uma vez que eles acedem a grandes conjuntos de dados estruturados e não estruturados, como por exemplo, a Web ou a grandes repositórios de informação provenientes de lá, de forma a obter as suas respostas, e no caso da comunidade de perguntas e respostas, fóruns online de perguntas e respostas em *threads* por temática. Os dados não estruturados fornecem um maior desafio, apesar dos dados estruturados de certa maneira limitar o leque de opções transformativas sobre os mesmos. A mesma disponibilização de tais conjuntos de dados de forma pública em formato digital oferecem uma maior liberdade para o público, e mais específicamente os investigadores das áreas específicas envolvidas com estes dados, permitindo uma fácil partilha das mesmas entre os vários interessados.

De um modo geral, tais sistemas não estão disponíveis para reutilização pública, porque estão limitados ao campo da investigação, para provar conceitos de algoritmos específicos, são de difícil reutilização por parte de um público mais alargado, ou são ainda de difícil manutenção, pois rapidamente podem ficar desatualizados, principalmente nas tecnologias usadas, que podem deixar de ter suporte.

O objetivo desta tese é desenvolver um sistema que colmate algumas destas falhas, promovendo a modularidade entre os módulos, o equilíbrio entre a implementação e a facilidade de utilização, desempenho dos sub-módulos, com o mínimo de pré-requisitos possíveis, tendo como resultado final um sistema de QA base adapaptado para um domínio de conhecimento. Tal sistema será constituído por subsistemas provados individualmente.

Nesta tese, são descritobos vários tipos de sistemas, como os de prospecção de informação e os baseados em conhecimento, com enfoque em dois sistemas específicos desta área, o YodaQA e o OAQA. São apresentadas também várias ferramentas úteis e que são recorridas em vários destes sistemas que recorrem a técnicas de *Text Classification*, que vão

desde o processamento de linguagem natural, ao *Tokenizatioin*, ao *Part-of-speech tagging*, como a exploração de técnicas de aprendizagem automática (*Machine Learning*) recorrendo a algoritmos supervisionados e não supervisionados, a semelhança textual (*Pattern Matching*) e semelhança semântica (Semantic Similarity). De uma forma geral, a partir destas técnicas é possível através de trechos de texto fornecidos, obter informação adicional acerca desses mesmos trechos.

São ainda abordadas várias ferramentas que utilizam as técnicas descritas, como algumas de anotação, outras de semelhança semântica e ainda outras num contexto de organização, ordenação e pesquisa de grandes quantidades de informação de forma escaláveis que são úteis e utilizadas neste tipo de aplicações. Alguns dos principais conjuntos de dados são também descritos e abordados.

A framework desenvolvida resultou em dois sistemas com uma arquitetura modular em pipeline, composta por módulos distintos consoante a tarefa desenvolvida. Estes módulos tinham bem definido os seus parâmetros de entrada como o que devolviam. O primeiro sistema tinha como entrada um conjunto de *threads* de perguntas e respostas em comentário e devolvia cada conjunto de dez comentários a uma pergunta ordenada e com um valor que condizia com a utilidade desse comentário para com a resposta. Este sistema denominou-se por MoRS e foi a prova de conceito modular do sistema final a desenvolver. O segundo sistema tem como entrada variadas perguntas da área da biomédica restrita a quatro tipos de pergunta, devolvendo as respectivas respostas, acompanhadas de metadata utilizada na análise dessa pergunta. Foram feitas algumas variações deste sistema, por forma a poder aferir se as escolhas de desenvolvimento iam sendo correctas, utilizando sempre a mesma framework (MoQA) e culminando com o sistema denominado MoQABio.

Os principais módulos que compõem estes sistemas incluem, por ordem de uso, um módulo para o reconhecimento de entidades (também biomédicas), utilizando uma das ferramentas já investigadas no capítulo do trabalho relacionado. Também um módulo denominado de Combiner, em que a cada documento recolhido a partir do resultado do módulo anterior, são atribuídos os resultados de várias métricas, que servirão para treinar, no módulo seguinte, a partir da aplicação de algoritmos de aprendizagem automática de forma a gerar um modelo de reconhecimento baseado nestes casos. Após o treino deste modelo, será possível utilizar um classificador de bons e maus artigos. Os modelos foram gerados na sua maioria a partir de Support Vector Machine, havendo também a opção de utilização de Multi-layer Perceptron. Desta feita, dos artigos aprovados são retirados metadata, por forma a construir todo o resto da resposta, que incluia os conceitos, referencia dos documentos, e principais frases desses documentos.

No módulo do sistema final do Combiner, existem avaliações que vão desde o já referido *Pattern Matching*, com medidas como o número de entidades em comum entre a questão e o artigo, de *Semantic Similarity* usando métricas providenciadas pelos autores

da biblioteca Sematch, incluindo semelhança entre conceitos e entidades do DBpedia e outras medidas de semelhança semântica padrão, como Resnik ou Wu-Palmer. Outras métricas incluem o comprimento do artigo, uma métrica de semelhança entre duas frases e o tempo em milisegundos desse artigo.

Apesar de terem sido desenvolvidos dois sistemas, as variações desenvolvidas a partir do MoQA, é que têm como pré-requisitos conjuntos de dados provenientes de várias fontes, entre elas o ficheiro de treino e teste de perguntas, o repositório PubMed, que tem inúmeros artigos científicos na área da biomédica, dos quais se vai retirar toda a informação utilizada para as respostas. Além destas fontes locais, existe o OPENphacts, que é externa, que fornecerá informação sobre várias expressões da área biomédica detectadas no primeiro módulo.

No fim dos sistemas cujo ancestral foi o MoQA estarem prontos, é possível os utilizadores interagirem com este sistema através de uma aplicação web, a partir da qual, ao inserirem o tipo de resposta que pretendem e a pergunta que querem ver respondida, essa pergunta é passada pelo sistema e devolvida à aplicação web a resposta, e respectiva metadata. Ao investigar a metadata, é possível aceder à informação original.

O WS4A participou no BioASQ de 2016, desenvolvida pela equipa ULisboa, o MoRS participou do SemEval Task 3 de 2017 e foi desenvolvida pelo próprio, e por fim o MoQA da mesma autoria do segundo e cujo desempenho foi avaliado consoante os mesmos dados e métricas do WS4A. Enquanto que no caso do BioASQ, era abordado o desempenho de um sistema de Question Answering na área da biomédica, no SemEval era abordado um sistema de ordenação de comentários para com uma determinada pergunta, sendo os sistemas submetidos avaliados oficialmente usando as medidas como precision, recall e F-measure.

De forma a comparar o impacto das características e ferramentas usadas em cada um dos modelos de aprendizagem automática construídos, estes foram comparados entre si, assim como a melhoria percentual entre os sistemas desenvolvidos ao longo do tempo. Além das avaliações oficiais, houve também avaliações locais que permitiram explorar ainda mais a progressão dos sistemas ao longo do tempo, incluindo os três sistemas desenvolvidos a partir do MoQA.

Este trabalho apresenta um sistema que apesar de usar técnicas state of the art com algumas adaptações, conseguiu atingir uma melhoria desempenho relevante face ao seu predecessor e resultados equiparados aos melhores do ano da competição cujos dados utilizou, possuindo assim um grande potencial para atingir melhores resultados. Alguns dos seus contributos já vêm desde Fevereiro de 2016, com o WS4A [86], que participou no BioASQ 2016, com o passo seguinte no MoRS [85], que por sua vez participou no SemEval 2017, findando pelo MoQA, com grandes melhorias e disponível ao público em `https://github.com/lasigeBioTM/MoQA`.

Como trabalho futuro, propõem-se sugestões, começando por melhorar a robustez do

sistema, exploração adicional da metadata para melhor direcionar a pesquisa de respostas, a adição e exploração de novas características do modelo a desenvolver e a constante renovação de ferramentas utilizadas Também a incorporação de novas métricas fornecidas pelo Sematch, o melhoramento da formulação de *queries* feitas ao sistema são medidas a ter em atenção, dado que é preciso pesar o desempenho e o tempo de resposta a uma pergunta.

Palavras-chave: *Question Answering*, *Information Retrieval*, Ordenação, Prospeção de Texto, Machine Learning

Abstract

Question Answering systems have been of great use and interest in our times. They are great tools for acquiring simple answers in a simple way, being of great utility in the area of information retrieval, and also for community question answering. Such systems have great potential, since they access large sets of data, for example from the Web, to acquire their answers, and in the case of community question answering, forums. Such systems are not available for public reuse because they are only limited for researching purposes or even proof-of-concept systems of specific algorithms, with researchers repeating over and over again the same r very similar modules frequently, thus not providing a larger public with a tool which could serve their purposes. When such systems are made available, are of cumbersome installation or configuration, which includes reading the documentation and depending on the researchers' programming ability.

In this thesis, the two best available systems in these situations, YodaQA and OAQA are described. A description of the main modules is given, with some sub-problems and hypothetical solutions, also described. Many systems, algorithms (i.e. learning, ranking) were also described.

This work presents a modular system, MoQA (which is available at `https://github.com/lasigeBioTM/MoQA`), that solves some of these problems by creating a framework that comes with a baseline QA system for general purpose local inquiry, but which is a highly modular system, built with individually proven subsystems, and using known tools such as Sematch, It is a descendant of WS4A [86] and MoRS [85], which took part in BioASQ 2016 (with recognition) and SemEval 2017 repectively. Machine Learning algorithms and Stanford Named Entity Recognition. Its purpose is to have a performance as high as possible while keeping the prerequisites, edition, and the ability to change such modules to the users' wishes and researching purposes while providing an easy platform through which the final user may use such framework.

MoQA had three variants, which were compared with each other, with MoQABio, with the best results among them, by using different tools than the other systems, focusing on the biomedical domain knowledge.

Keywords: Question Answering, Information Retrieval, Text Mining, Natural Language Processing, Machine Learning

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent times, Question Answering (QA) systems have attracted great interest in the information retrieval community (the activity of obtaining information resources relevant to a query from a collection of information resources), and also in the community, QA (cQA) [40]. These systems make effortless all the human work that would be necessary to acquire the answers wanted in the first place. The answers and knowledge the users want and require are more often on the Web while interacting even with other users in order to acquire knowledge, such as in forums. Another case these systems got much attention was by the performance of the IBM Watson [29] system on the show Jeopardy!. Furthermore, a characteristic of QA (more precisely cQA), is that a user resorts to the web for answers without a given structured knowledge base. The arbitrariness of cQA forums, the dependence and waiting time on their results, may slow the obtainment of answers in real time. Also, public forums are dependent on the users' input (i.e. answers), which might be rather unstructured, not straight to the point, related to the question at hand, not well written (i.e. grammatically), lengthy or even not correct. Moreover, the answers are subject to interpretation, and a definite way of showing correctly the answer might help the user to reach a correct conclusion. [25, 39, 2, 105]

The QA Research is now proceeding in several semi-independent tiers depending on the precise task formulation and constraints on the knowledge base and new researchers entering the field are focusing on variously restricted sub-tasks as no modern full-scale software system for QA has been openly available until recently.

1.2 Problem

The question answering problem is considered to be a function of unstructured user query that returns the information queried for. It is a harder problem than a linked data graph search, which requires a precisely structured user query, or even a generic search engine,

which returns whole documents or sets of passages instead of the specific information. Such a task is a natural extension of a search engine, as in Google Search or personal assistants like Apple Siri, or even the highly publicized IBM Watson Jeopardy!.

Many QA system architectures have been proposed in the last 15 years, applying different approaches to Information Retrieval (IR). A common restriction of the QA problem concerns selecting the most relevant snippet (a small piece or brief extract) that may contain the answer, given an input question and set of pre-selected candidate snippets. The answer selection task is certainly worthwhile as a component of a QA system but it is not a complete system by itself. It may be argued that returning a whole passage is more useful for the user than a direct narrow answer, but this precludes any reasoning or another indirect answer synthesis on the part of the system, while the context and supporting evidence can be still provided by the user interface.

In spite of systems having great performance, most of the time these types of systems are not available for public reuse, they are merely research prototypes, to achieve presupposed results and/or scores, and proof-of-concept systems of specific algorithms. Some systems often share the same issues they try to address, such as feature use depending on the question/challenge at hand, the value of such features, in which their final impact may be close to null. The focus of such systems, usually is not to provide researchers with a system, tools or materials that would be used and shared by a larger public, which could better serve their researching purposes. [39, 50, 31]

The few systems available, are not often reusable, in spite of theoretically being so, when implementing or trying to increment or changing the domain of those systems, turns into a burdensome task that includes reading a lot of documentation and depending on the developers' availability and understanding to help those that are trying to reuse their systems. Just to configure such a system, by itself is a very complex task, that depends heavily on the users' knowledge of such systems and configurations, which includes a long list of prerequisites. YodaQA [10] and OAQA systems [117], which were the main focus of research and inspiration of this work, are later highlighted and commented.

Specific search engines, such as PubMed, do not fully address the semantic question and answering problem, which recur to large biomedical repositories, using domain knowledge, among other resources. Moreover, usually, there is no combination of the sources, which seldom return texts or structured information, adding, even more, work to the researcher to deal with. On the other hand, QA systems as a whole focus on delivering concise answers, using, for example, semantic indexing, which unfortunately is frequently implemented based on human intervention. [100, 48] Other forms of less accessible knowledge, are the limited data diversification of knowledge bases, even when combining datasets, and even though a certain question might not be answerable, which is by itself another problem. There are also systems which are restricted to the types of answers they may answer, such as factoid, list, yes/no or even summary questions. For

example, YodaQA only answers in a factoid fashion. A factoid answer consists of a small word expression which is accepted as fact; a list answer is a collection of factoid answers; a summary answer is a brief statement or account of the main points of the question.

Finally, different forms of questioning imply the use of SPARQL (SPARQL Protocol and RDF Query Language, a semantic query language for databases stored in Resource Description Framework (RDF) format) queries, which involve another step for comprehension and use of the user. There is some research in order to ease this step, by using template based approaches, which limit the types of question one can make, to template-free which is the first approach given. However, the template-free mindset requires additional effort to cover every possible graph pattern. [40]

1.3 Objective

This thesis aims at addressing and providing an initial solution to some of the problems presented, by creating a framework that comes with the tools and modules for a baseline QA system. This modular system, built with individually proven subsystems, is also to be effective, due to combinations of those modules, hereby introducing the Modular Question Answering system (MoQA). The purpose is to balance performance performance, prerequisites, and the ability to edit those modules for personal use. In addition, have such a system available locally, with the minimum access to external resources or tools, with as few prerequisites as possible. In a sense, start closing the gaps on the problems of important existing QA systems described in the next chapter. In sum, to make a seamless framework for users to ask questions and retrieve answers about the documents they make available to the system, with the help of an annotated corpus, if provided.

Through the use of Machine Learning techniques, to make available an answer classification module that distinguishes good and bad answer candidates for given questions. Each question is analyzed, in which such things as keywords are obtained, related documents are retrieved, and a list of answer candidates is returned. With such a framework, show a proof-of-concept for the biomedical area, by building a QA system based on the PubMed, which comprises more than 26 million citations for biomedical literature from MEDLINE, life science journals, and online books.

In summary, to have a general, simple to use the system for those that want to ask questions about a given dataset, while being modular, so that experienced users may change specific modules to achieve better results for their own purposes. In this case, the retrieval system would search for relevant documents linked to other questions in order to find potentially appropriate answers, assessing the usefulness of the retrieved documents with respect to the question posed by the user.

Hypothesis It is possible to build a modular system from a predefined framework, with reasonable performance, a smooth configuration for users at a beginner level of cod-

ing knowledge. Such framework is to be adaptable to distinct domain knowledge. The systems' capabilities include indexation and search of large quantities of documents, annotation, ranking and classification, with an interactive web interface.

1.4 Contributions

This system is the result of work being developed since February 2016, which took the idea from WS4A (Web Services For All), where a lightweight system was built using mostly web services and also local semantic analysis, which got the second place in exact and ideal answers for the first batch of the BioASQ 2016. [86]

The system, renewed and with some new ideas, has successfully participated in the SemEval Task 3, more precisely Subtask A, achieving top results for the development set of 2017 and test set of 2016 [85]. These results were made available through the organization's scorer and published in the following repository `https://github.com/migueljrodrigues/MoRS-Scores`.

Finally, a use case is materialized by making a second iteration (after WS4A and MoRS), dubbed MoQABio (a variation of MoQA), achieving improvements of at least 22x in the mean of the results in the categories rated: documents and snippets, achieving top 3 results in these categories, comparing to last year's results.

Demos for the MoQA system are available for the public at `https://github.com/lasigeBioTM/MoQA`, for those that wish to contribute and also reuse for their own purposes, along with instructions to run the demos.

1.5 Document Structure

This document is organized as follows:

- In Chapter 2, is made a review of Text Mining techniques, of tools that use some of these techniques, the most used datasets in the kind of task this thesis focused on, along with ranking algorithms. Finally, an overview of the types of QA systems is given, along with varied examples, focusing on two major systems: YodaQA and OAQA.

- In Chapter 3, a history of development is told, from last year's work, WS4A, to MoRS, finalizing with MoQA and its variants.

- In Chapter 4, the results of such development are shown, along with comparison among the tasks from which the data was used, as well as cross-comparisons between the developed systems.

- in Chapter 5, the last chapter, conclusions are made, with reflections, thoughts and suggestions on some future improvements to be made in the future.

1.6 Planning - an aftermath

This thesis started on 2016-10-15, and every step of planning was successfully accomplished in time, from the research phase (related work and tools), to the participation in SemEval 2017 (development of MoRS and article writing), to the development of MoQABio.

Chapter 2

Related Work

In this Chapter, I intend to contextualize the work developed, main technologies and algorithms, with finally presenting an overview of the current state of the art on biomedical question answering systems. The basic concepts in this domain will be described to fully understand the work developed, with a special focus on the text mining techniques and tools that were used in this thesis.

2.1 Text Mining

Text Mining has a big role in this kind of system, focusing on smaller portions of the information contained within plain text documents. It differs from similar tasks like Information Retrieval, Text Summarization (TS) and Natural Language Processing (NLP) where the focus is on the document as a whole, with a bigger level of granularity) [20].

2.1.1 Techniques

Several techniques can be applied for addressing the tasks previously described. In the following subsections, the most used techniques are described, focusing the ones that are used in this work and applied for the biomedical domain.

2.1.1.1 Natural Language Processing

Natural Language Processing is an area of computer science that focuses on processing texts written by and for humans. Some techniques used for addressing NLP tasks are also employed in text mining, such as the recognition task. Although both the NLP and text mining fields are related to the processing of narrative texts and thus share techniques, these two fields are different from each other. For instance, NLP tasks are focused on processing larger text files (i.e. a whole document), while text mining concerns more about the detailed information contained in the documents [20]. Text mining commonly uses NLP techniques to parse the input text into a machine-readable form [92]. The

following NLP techniques are some of the most commonly used in text mining systems, and they are also broadly applied in the biomedical domain:

2.1.1.2 Tokenization

The first technique is tokenization, where the text used as input must be split into units called tokens, so it can be processable by a machine. Although these tokens are usually single words, they may also consist of numbers, symbols or even phrases. To retrieve these tokens from the text, a tokenization parser must be employed. This parser splits the input text based on a set of predefined rules. A simple approach would split tokens according to a group of pre-defined delimiters symbols (i.e. spaces, dots, commas), however, this naive approach obviously does not achieve the best results. Depending on the domain text and structure type, more advanced heuristics must be applied in order to improve the quality of the tokens retrieved from the text (e.g. deciding when quotes or brackets are parts of the word). The Stanford Tokenizer [61] and Banner [52] are two examples of systems developed specially for text written in English. Although this is a common and rather simple first step in text processing systems and specifically QA systems, the wrong implementation of this process may lead to a poor performing system [108].

2.1.1.3 Part-of-speech tagging

For each word in the text, a part-of-speech tag is assigned to identify nouns, verbs, adjectives, etc. Since the same word may belong to distinct classes, the label is assigned based on the definition of the word itself along with the context. The Stanford NER Part-Of-Speech Tagger [98] and the GENIA Tagger [102] are implementation examples.

2.1.1.4 Machine Learning

Machine Learning (ML) is a scalable and quite flexible solution that learns through examples observed from previous cases and generates a model capable of resolving new future cases [3, 20, 82, 111].

Two types of corpora can be used as input for the ML algorithm: labeled, also known as annotated, and unlabeled corpora. The existence of these documents is essential to the success of any recognition and normalization system.

Machine learning algorithms can be divided into two categories: supervised algorithms and unsupervised algorithms, according to the type of documents used as input.

Supervised Algorithm Supervised algorithms require, in a first instance, the training of a classification model based on the annotated data received as input. This model can then be used to accomplish an automatic classification task. Supervised machine learning algorithms are the most common approach used by QA systems for question answering

tasks in the biomedical domain. As this approach requires biomedical experts to manual annotate a set of documents to be used as training data, a considerable effort is intrinsically associated with this approach. For this reason, participation in competitions is a solution such as BioASQ and SemEval [100, 48, 70], since the organizers provide the supervised training set of considerable sizes. The supervised algorithms are based on the definition of a defined group of features that will be used to represent the training data. This representation is processed by the machine learning algorithm which generates a model based on that information [37]. In order to not end up with an overfit or bland model, both the features and input knowledge used have to be carefully chosen and previously researched. Several supervised machine learning algorithms can be applied, is the following the most common in biomedical domain:

- **Association Rules**: based on the annotated data, rules are generated by identifying frequent patterns within the corpora. Generated rules can be as simple as if A and B then C [112].
- **Decision Trees**: based on the features retrieved from the training data, a decision tree is generated. The tree is composed of nodes which represent a condition, links which connect different nodes and leaves which represent classes. Following the decision tree from the root, several conditionals nodes will define the branch to follow and thus the class to be assigned. This type of algorithm is easier to understand but it easily gets overfit to the training set [4].
- **Support Vector Machines (SVM)**: the features are represented as points in a vector space. Input instances are mapped to a high-dimensional feature space where a linear decision model is constructed. The model generated is a spatial representation of the training data, clustering similar categories with the largest gap possible between other distinct categories clusters. [22]
- **Conditional Random Fields (CRF)**: Statistical models that create a sequence segmentation of classes, based on the training data. The statistical model will assign the most probable class to each token. In this learning algorithm, the context is taken into consideration for choosing the right classification class [49].

On supervised algorithms, the annotated data must be pre-processed by using some of the techniques from Natural Language Processing area. During pre-processing, tokenization is the most relevant one, transforming the input text into a set of individual tokens. For each of these tokens, a specific class is assigned, which have a semantic meaning associated.

Unsupervised Algorithms On the other hand, unsupervised machine learning algorithms are based on unlabeled data, and therefore no processing is required to set up the

input data. These algorithms are used to detect structures and patterns within the input data. The most commonly used unsupervised algorithm within text mining is clustering, that implies the creation of clusters [75]. A cluster is a group of entities with similar features, therefore, each cluster represents a set of unique entities as a whole. Brown clusters are one the most known clustering algorithm, used for grouping related words. According to Brown [14], this technique allows reducing the sparsity of the data, generating a lower-dimensional representation of the unlabeled data used as input.

2.1.1.5 Pattern Matching

Pattern matching algorithms can be employed, retrieving the most similar concept from the knowledge base (i.e. a set of documents), based on the similarity of two given descriptors. These algorithms are based on the assumption that similar strings are related between them, and were developed for addressing this problem by calculating the distance between two strings and therefore their similarity. The similarity, in most of the cases, normalized between the values 0 and 1, being the first a result when two strings are distinct and the second when they are totally similar. Some distinct algorithms are available [19]:

n-Gram n-Grams are sub-strings of size n retrieved from a string to compose a longer one. For instance, the entity ”eye” contains the bigrams ’ey’ and ’ye’, and the trigram ’eye’.

The distance between two strings consists of counting the frequency of n-grams that two strings have in common. More n-grams in common means that the strings are more similar to each other. The similarity measure is obtained by dividing the number of n-grams in common by the number of n-grams in the shorter string (a.k.a. Overlap coefficient), the union of n-grams in both strings (a. k. a. Jaccard similarity) or even by the average number of n-grams in both strings (also known as Dice coefficient) [19, 47]. This algorithm definition allows to compare two strings at character-level (Levenshtein distance) but also allows to perform the comparison with larger windows including a sequence of characters (n-grams).

Cosine-Similarity This algorithm computes the similarity between two vectors of attributes. Vectors that are similar with each other, will also be close to each other in the space model and thus, higher similarity [32]. The similarity value is calculated using the inner product space of the vectors.

Considering a string as a document representing it in a vector space model one may employ information retrieval techniques and use this algorithm to retrieve a measure of similarity between two strings.

2.1.1.6 Semantic Similarity

Given two concepts from a specified ontology, semantic similarity measures can be applied to return a value which represents the closeness in meaning between those concepts, or their distance. This measure allows one to compare the similarity between two concepts, assuming that concepts with high semantic similarity will more likely be related. The semantic similarity algorithms are intimately related to the information content of the compared concepts. The following algorithms are the most common:

2.1.1.7 Community Question and Answering Features

In specific cases such as cQA, there are some more features to be used, like: **(1)** the number of question marks in the documents, **(2)** whether it contains smileys, e-mails, "thank you" phrases, **(3)** number of offensive words from a predefined list, **(4)** length of the answer (in characters), **(5)** if it includes a first person singular, or **(6)** plural pronoun, or even **(7)** if the author of the comment is the same as the author of the question at hand. [65]

Moreover, other characteristics are also used such as the comment's position in the thread, and the ID of the author of the comment, if available. After tokenization, another metric could be the ratio of the comment length and of the question length (in of a number of tokens), the number of comments from the same user the thread and the order in which they are written by him. Another aspect of metadata worthy of exploration is the presence and the number of links in the question and in the comment (inbound or outbound), taking into consideration that the presence of a reference to another resource is indicative of a relevant comment. [68]

2.1.2 Summary

After this section, one may see a varied Text Mining techniques that serve purpose of providing more information or transformation as to the text given. These techniques are from NLP, to tokenization, POS tagging, to Supervised and Unsupervised ML to Pattern Matching and Semantic Similarity (with many measurements to choose from.

2.2 Tools

In this section, a myriad of tools will be described, along with their purpose in the development of QA systems. The tools serve different purposes and use techniques such as the one's described in the previous section.

Machine learning implementations will be especially focused, due to their popularity on existing text mining solutions, as well as for the biomedical domain. Several machine learning implementations are available and although some of them were developed for a

specific domain, they can be adapted for other domains when using the appropriate corpus as input. In this section, some of the most commonly used software solutions are briefly described.

2.2.1 Annotation

In this section a list and respective description of text annotation tools will be given, along with their capabilities and some possible use cases.

2.2.1.1 Stanford NER

Stanford NER (Named Entity Recognition) is a Java implementation of a Named Entity Recognizer and an open source software which implements a machine learning algorithm by employing a linear-chain conditional random field (CRF) approach for building probabilistic models based on training data. The tool uses a supervised approach since it leverages on the existence of annotated data.

Stanford NER was developed specially for the recognition of general domain entities, such as person and company names, or gene and protein names, etc. It allowed an error reduction of 9 percent from the previous state of the art systems. It allows the definition of a set of features to be used for the model training, like the addition of unsupervised approaches (clusters) and normal token features such as the window size (number of tokens which defines the local feature windows), token shape, token Part-of-Speech, among others. Although Stanford NER was not specially developed for the biomedical domain, it can be applied to this specific domain and achieve acceptable results.

2.2.1.2 Natural Language Toolkit

The Natural Language ToolKit (NLTK) is a primary platform for building Python systems to work with data consisted as human language. It provides easy-to-use interfaces to many corpora and lexical resources such as WordNet, along with a group of text processing libraries for tasks such as classification, tokenization, stemming, tagging, parsing, etc.

Due to extensive documentation, NLTK was made suitable for linguists, engineers, students, educators, researchers, and industry users, being made available for Windows, Mac OS X, and Linux. NLTK is a free, open-source, community-driven project.

2.2.1.3 GENIA Tagger

GENIA Tagger serves for building programs which may need part-of-speech tagging, shallow parsing, and named entity recognition for specifically for biomedical text. [1]

The GENIA tagger takes sentences in English, analyzes them, and outputs the base forms, part-of-speech tags, chunk tags, and named entity tags. The tagger is specifically

[1] http://www.nactem.ac.uk/GENIA/tagger/

tuned for biomedical text such as MEDLINE or PubMed abstracts. This tagger may be useful processing if one needs to extract information from biomedical documents.

As one would infer, general-purpose POS taggers do not perform as well as biomedical POS taggers biomedical text because lexical characteristics of biomedical documents are considerably different from those of other texts such as newspaper articles, which are often used as the training data for a general-purpose tagger.

The GENIA tagger is trained with data from the Wall Street Jornal corpus, from the GENIA corpus and the PennBioIE corpus, ending up with working on various types of biomedical documents.

2.2.2 Semantic similarity

In this subsection, tools that provide and ease semantic similarity measurements will be highlighted, described and be shed some light upon examples of use of them, along with their support.

2.2.2.1 Sematch

Sematch is a tool that provides many similarity measures, that include semantic similarity scores between concepts, words and entities, focusing on specific knowledge-based semantic similarity metrics which rely on structural knowledge in taxonomy, and statistical information contents.

Sematch offers a useful way for research purpose in a small dataset to compute entity similarity and relatedness, being advisable the storage of results locally to avoid performance issues. [120]

Some of these measurements are described in the following paragraphs.

Word Similarity The Sematch word similarity is computed based on WordNet along with various semantic similarity metrics, extending the NLTK's WordNet and other semantic similarity metrics. NLTK provides path [79], lch [51], wup [113], res [84], lin [55], jcn [42], li [54] and wpath [119].

The multilingual word similarity is supported by Open Multilingual WordNet, providing support for various languages.

DBpedia Concept Similarity As for DBpedia Concept (i.e.`http://dbpedia.org/ontology/Actor`) Similarity, it was implemented a class with the semantic similarity metrics used in WordNet taxonomy, making it possible to parse DBpedia concept data to Taxonomy and ConceptSimilarity for computing semantic similarity between DBpedia concepts.

DBpedia Entity Similarity Two entity (i.e. `http://dbpedia.org/resource/Madrid`) similarity measurements are provided, **(1)** entity relatedness which is based on DBpedia link association (mainly measures entity link overlap between entities) [67], with the other being **(2)** entity similarity based on YAGO concept similarity [64], which is another kind of measurements not used in the Work.

The similarity method needs to run two SPARQL queries to obtain the required features (entity concepts (e.g. movie, actor)).

2.2.2.2 DiShIn

In addition to Sematch, DIShIn (Semantic Similarity Measures using Disjunctive Shared Information) is a software package which provides several functions regarding semantic similarity measurements from a rdf or owl file. Some functionalities include the production of the semantic-base as a SQLite database, the score for semantic similarity based on teh SQLite database, examples of semantic similarity for metals and ChEBI, Gene Ontology (GO) and Human Phenotype ontology (HPO) ontologies.[23, 24]

2.2.3 Searching and indexing

In order for one to store and manage a large set of documents or table entries, one may need some tools that help in such a task, as the all regular databases, with MySQL being one quite common example. In addition, there are also tools that make it easy to search among these databases. Finally, there are those tools that merge searching and indexing, and in this section, such tools are described and compared among one another.

2.2.3.1 Lucene

Lucene [63] is one software package including all the pattern matching algorithms previously described. This package allows the creation of index words from dictionaries based on a given pattern matching algorithm. This results in an easy and fast search for the most similar entities according to a specifically chosen algorithm and within an indexed dictionary. Lucene is a high-performance, full-featured text search engine library written entirely in Java.

2.2.3.2 Solr

Solr [91] is a popular, open source search platform from the Apache Lucene project written in Java. It includes as features a full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document (i.e. Word, PDF) handling, and geospatial search. It is highly scalable, providing distributed search and index replication, powering the search and navigation features of many large internet sites.

Solr uses the Lucene Java search library at its core for full-text indexing and search and has somewhat REST and JSON APIs that make it accessible for usage in many programming languages.

2.2.3.3 Elasticsearch

Elasticsearch [36] (ES) is an increasingly popular search engine based on Lucene and developed in Java, which provides a distributed, full-text search engine with a web interface and schema-free JSON documents. Elasticsearch is the most popular enterprise search engine followed by Apache Solr, also based on Lucene.

Elasticsearch can be used to search all kinds of documents, whilst having a scalable search with near real-time search, and supports multitenancy. Elasticsearch is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards, and acts as a coordinator to delegate operations to the correct shard(s).

Ranking algorithm This algorithm altogether is part of Lucene, which runs under the hood of Elasticsearch, its own algorithm named Practical Scoring Function. This is a similarity model based on Term Frequency (TF) and Inverse Document Frequency (IDF) that also uses the Vector Space Model (vsm) for multi-term queries.

The TF/IDF takes into account term frequency (how often the term appears in the field), inverse document frequency (how often each term appears in the index) and field-length norm (how long is the field).

API Elasticsearch uses Lucene and tries to make all its features available through the JSON and Java API.

2.2.3.4 Solr vs. Elasticsearch

In spite of both tools having great utility, and being very similar in their results, some specific characteristics distinguish them.

Elasticsearch was developed in the age of REST APIs, so it is more aligned with web developers' mindsets.

Elasticsearch's Query Domain Specific Language (DSL) syntax is flexible and rather easy to write complex queries with it, though it may result in verbose queries. Solr does not have an equivalent.

Elasticsearch's documentation is in constant change and update, it is a difficult task to have updated documentation since constant updates keep bringing new functionalities to the tool. On the other hand, Solr is quite consistent and really well-documented.

Elasticsearch is an out-of-box system, but that may make users negligible when configuring and getting to know the system according to their necessities, ending up with

problems in production. With Solr, since the configuration is done through a JSON or YAML file, this file may end up to be quite long and confusing. While Solr has adapted itself to accept JSON as a format of communication after some releases, Elasticsearch has been ”thinking” and working with JSON all along.

Elasticsearch does not need an external service to maintain information, naming, provide distributed synchronization, and group services. As researched, the choice tends to go to what programming languages developers prefer: for Java/C# developers, got with Solr. If one likes web developing languages such as in Javascript or Ruby, one should probably go with Elasticsearch.

All in all, ES and Solr similar feature-parity and from a feature standpoint, no tool is better than the other, unless on a personal choice of development. Performance-wise, they are also quite similar, although ES' recent autocomplete implementation, is a dramatic departure from previous Solr implementations, producing also faster responses at scale. Alas, the tiebreaker is the difficulty that one wants to have in configuring the system and how specific/complex the system is to be.

2.2.4 Summary

All in all, there are many tools available for developers to use, with a restrict and focused set presented in this section. From biomedially trained annotation tools to those with a more general annotation, semantic similarity measurements, which include ontologies, to tools that help developers organize and search through large databases of documents with ease.

2.3 Datasets

Two types of corpora can be used as input for the ML algorithm: labeled (or annotated) and unlabelled. Unlabelled documents consist of raw information available regarding the domain associated with the task, such as a set of raw clinical notes, i.e. notes that were not annotated or evaluated by any expert. This data is less expensive and has a higher level of availability than the labeled data since no effort is required by experts. On the other hand, labeled documents, are an essential resource for the production of high-performance systems (i.e. classification systems). Labeled documents consist of documents containing the relevant information identified in the text (also known as golden standard/documents), is usually generated by experts manually in the domain and thus, a great effort is required. This means that not only the data is more expensive, that these documents are less available, or even in smaller quantities.

There are many datasets available for training in many areas and also for many of the modules. Some of the data collections made available by [53], contain all the data used in various learning question classification experiments, which has question class definitions,

the training and testing question sets, examples of preprocessing the questions, feature definition scripts and examples of semantically related word features.

All algorithms demand the existence of a corpus to be used as knowledge base input. Without it, it is impossible to implement machine learning algorithms. In this section, the main corpora and datasets used in biomedical text mining are described next.

2.3.1 PubMed

PubMed is a free access article database created in 1996, containing more than 27 million citations for biomedical literature from life science journals, online books and MEDLINE, which is one of the primary components. Several fields from the biomedical domain are incorporated in this database such as medicine, health care system and preclinical sciences. The PubMed database can be seen as a free access extension of the MEDLINE database since this last represents the largest subset of the PubMed [90].

2.3.2 MeSH

The articles within MEDLINE/PubMed are indexed with specific keywords according to the Medical Subject Heading (MeSH), a structured database [60]. MeSH is composed by terms naming descriptors where the root is more abstract and the leafs are more specific, allowing one to search for articles within MEDLINE/PubMed. MeSH is not an ontology since the structure is based on links and not semantic relations as expected in an ontology. The 2014 MeSH release was composed by 27,883 descriptors. [57] MeSH is an essential tool to perform relevant and more complex searches through all available references in the MEDLINE/PubMed database, allowing one to research the most relevant sources about a given subject through a set of queries.

2.3.3 Open PHACTS

Open PHACTS (Open Pharmacological Concept Triple Store) is a public–private initiative between numerous academic and commercial organizations, founded on semantic web and linked data principles, that has the objective of providing better, cheaper and faster drug discovery. Its Discovery Platform was developed to tear down barriers to drug discovery in the industry as well as academia. [110]

It contains data sources include: ChEBI, ChEMBL, SureChEMBL, ChemSpider, ConceptWiki, DisGeNET, DrugBank, Gene Ontology, neXtProt, UniProt and WikiPathways.

The data within the platform is available in many formats (i.e. JSON and XML) in order to suit as many applications as possible.

2.4 Ranking

In these last years, learning to rank algorithms have been applied to information retrieval tasks. As seen in the previous section, they aim at learning a model that given a query and a set of relevant documents and find the appropriate ranking of given documents according to the relevance of its features. There have been much learning to rank methods developed over the years, which can be divided into three main categories: pointwise methods, pairwise methods, and listwise methods.

2.4.1 Pointwise ranking

Pointwise methods treat the ranking problem as a standard classification or a regression task. An example of use would be in IBM's Watson, which uses logic regression to score the relevance probability.

The work of [11] presents a general rank-learning framework for passage ranking within QA systems using linguistic and semantic features. Constraints are composed of a mixture of keyword and named entity features, as well as features derived from semantic labeling. They have shown that a trained ranking model using a rich feature set achieves greater than a 20% improvement in Mean Average Precision over baseline keyword retrieval models.

2.4.2 Pairwise ranking

On the other hand, pairwise methods like FRank, SVMRank, RankNet, RankBoost aim to learn the pairwise preference of candidate answers rather than their absolute rank.

LambdaRank, is inspired by RankNet, is a pairwise ranking method. It is based on the idea that in order to learn a model, the actual value of the loss function is not needed, only the gradient of the loss function is enough. Once a gradient is known, it can be used with standard optimization methods. [11] LambdaRank has shown in [1] to perform very well in ranking tasks.

2.4.3 Listwise ranking

Listwise methods operate on the entire list of candidate answers. Unlike pointwise and pairwise methods where a loss based on the rank of the individual candidate answer is minimized, in listwise methods, a direct loss is minimized between the true ranks of the list and the estimated ranks of the list. These methods have outperformed the other two types of methods for information retrieval tasks. Many of these methods allow for the optimization of the final metric relevant to the application. Examples of listwise methods are LambdaRank, AdaRank [114] and ListNet [17].

2.4.4 SVMrank

SVMrank is a sub-module of SVMstruct [44] which features: a fast optimization algorithm, a working set selection (based on steepest feasible descent), the ability to handle thousands of support vectors and training examples.

A set of classified arrays are transformed in the following format, where the first digit is the importance/relevance of that comment. The larger the first number is, more important is the comment, while qid denotes the question number, for classification within that question, and the 1:, 2:, 3:, etc. are the scores for each feature. Here follows an example:

```
1 qid:2 1:0 2:0 3:1 4:0.2
2 qid:2 1:1 2:0 3:1 4:0.4
1 qid:2 1:0 2:0 3:1 4:0.1
1 qid:2 1:0 2:0 3:1 4:0.2
```

2.5 State Of The Art

Since the beginning of the computational era, the question and answering systems have been an important part of this area of investigation. Such research has been registered since the early 1960s, where maybe the two most important standards have been implemented throughout the years, IR and knowledge-based question answering systems.

There is already a great number of biomedical question answering systems described through their respective publications, which use different approaches and purposes, which shows that this area is growing at an increasing rate. Well-known conferences, such as the Association for Computer Linguistics (ACL), confirm this same interest and growth in this area of publication.

In such systems, mostly text mining techniques are employed in order to automatically retrieve the knowledge within narrative text. Although each domain has its specific purposes and technicalities, they share the same goal which consists on retrieving the knowledge contained in narrative documents and present useful answers and support documents.

Most of the recent work developed in this domain is based on machine learning and deep learning approaches, a more time-solid solution which consists of the application of artificial intelligence algorithms.

Nowadays, systems like IBM's Watson have reached state of the art results. Such systems have been focusing on factoid questions, which are questions with simple facts of short text expressions. Current answers may vary between names, temporal expressions or even locations. There are some kinds of question answering systems, and during some deep research, many were found, which are described in the following areas.

In this section, the most prominent and inspiring systems will be described, along with their main characteristics and uses.

2.5.1 Information Retrieval Factoid systems

One kind of QA systems is the IR with factoid answers, a type of answer which MoQA will also present, with the addition of retrieving information from specific documents. This is why information retrieval systems are important to MoQA.

The IR-based question answering system relies heavily on the amount of information available in text form on the Web, or even in specialized collections such as MEDLINE/PubMed. The system would use various techniques to extract useful information from the documents available, according to a given question. The method most used starts with determining the answer type required and then ”questions” are formulated and processed. Ranked documents and their respective excerpts are then returned in an orderly fashion. Knowledge-based question answering includes also a semantic representation of the query made.

Independently of the chosen representation, a variety of databases may be used, based on Structured Query Language (SQL), or even triple stores (purpose-built database for the storage and retrieval of triples through semantic queries) like Freebase [13] or DBpedia [6]. IBM's Watson, mentioned before, has a hybrid approach, using text datasets and structured knowledge bases. DeepQA, the system inside Watson, extracts a wide variety of meanings from the question and then searches for candidate answers in both types of databases. Each candidate answer is scored using a variety of classification criteria.

The objective of the question processing phase is to extract a number of varied pieces of information from the question. The answer type specifies the kind of entity the answer consists of. The query specifies the keywords that should be used for the IR system to use when searching for documents. By knowing the answer type of a question and some characteristics of the same, one may avoid looking at every sentence or noun phrase in the entire set of documents for the answer.

2.5.1.1 Question classification

Question classification (i.e. pattern recognition in questions) accuracy is often high on easy question types such as 'person', 'location', and 'time' questions', while others like 'reason' and 'description' are much harder. Although the set of documents is generally ranked by relevance, an analysis of the results of varied systems showed that the top-ranked document is probably not the answer to the question. This happens because documents are not an appropriate unit to rank when it comes to the goals of a question-answering system. It is considered that a highly relevant document that does not decisively answer a question is not an ideal candidate for further processing. The next stage proposed consisted in extracting passages from the retrieved documents. This has been the case of systems submitted for SemEval of 2016. [70]

2.5.1.2 Snippet retrieval

In the passage retrieval stage, first passages are filtered out in the documents that do not contain potential answers and then rank the rest of the documents according to their likeliness of containing an answer to the question. The remaining passages are then ranked, most of the time using supervised machine learning (inferring a function from labeled training data), with a small set of features extracted from the documents. Such ranking of passages regarding a question was a task approached by MoRS, and will be described later on in the developed work.

2.5.1.3 Answer extraction

There is a clear difficulty when it comes to 'definition' questions, and in such case, handwritten regular expression patterns are used to help extract the answer.

An example of such a system is the one in [105], which answers users' questions submitted to the Yahoo! Answers website that has not been previously answered by humans. In this system, sets of keyword queries were generated given the title and body of the question, and then a collection of web pages would be retrieved taking them into account. Answer candidates are then extracted from web pages and then ranked, all this in about a minute.

2.5.2 Knowledge-based systems

In Knowledge-based systems, the information structure is quite the opposite from the one present in IR systems, where information is in a structured form in the first and in an unstructured form in the latter, with some small exceptions. The term knowledge-based question answering for a system that answers a natural language question by mapping it to a query over a structured database, with defined rules of structure.

These systems typically use the RDF paradigm and accessible via SPARQL. The QA problem can be then rephrased as learning a function translating free-text user query to SPARQL query (or a lambda expression).

When in presence of a set of questions and correspondent logical (also known as supervised data), it is usual to take those pairs of training tuples or triples and produce a system that maps from new questions to their logical forms. These systems generally bootstrap by having a small set of rules for building this mapping, as well as an initial lexicon.

Supervised learning cannot cover such a wide variety of forms that, for instance, factoid questions can take, making it difficult for training sets to be labeled correctly. [45]

2.5.3 YodaQA

The first major system in which MoQA takes ideas is from YodaQA [10], and more specifically its pipeline, which is implemented almost in its entirety in Java, not a common language among lay researchers, while using the Apache UIMA framework, a widely known choice among researchers of the area. The Apache UIMA is an Unstructured Information Management application licensed by Apache, which consists of software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user.

With the YodaQA system, the authors seek to reunite and boost research efforts in Question Answering, providing a modular, open source pipeline for this task, while allowing integration of various knowledge base paradigms, answer production and analysis strategies and using a machine learning models to rank the answers. The authors got inspiration from other systems such as DeepQA.

The authors kept an interest in a general purpose QA system, considering an "open domain" general factoid question answering, rather than domain-specific applications, though flexibility is kept in this direction as one of the goals.

While the task becomes harder, the YodaQA authors believe their system is a universal system that could be readily refocused on a specific domain or proprietary knowledge base.

This system is largely influenced by the DeepQA model of IBM Watson, and includes the following steps [78]:

- Question Analysis, where natural language features are extracted from the input and representations of the question are produced
- Question Analysis extracts natural language features from these representations
- Answer Production generates a set of candidate answers based on the question, by performing a Primary Search in the knowledge bases by using Passage Extraction and Analysis
- Answer Analysis generates answer features based on detailed analysis
- Answer Merging and Scoring removes duplicates and machine learned classifier to score answers according to their features

For an already installed YodaQA system, there is also a resourceful REST (a stateless, client-server, cacheable communications protocol) API endpoint available from a local server, accessible for querying from the user.

The objectives of YodaQA included: **(1)** providing an end-to-end, universal pipeline integrating different knowledge base paradigms in a modular way, and **(2)** one which is domain flexible.

YodaQA presents some failures since it is cumbersome to install and configure, or even to make modifications to its initial code in order to adapt the system to specific domain QA systems, as seen in the system's GitHub page, and nowadays without a working online demo. The fact of being only of factoid answer presentation is rather limiting, and those factoid answers are at times not written in a correct fashion. It is also a very complex and heavy system for the CPU and requires some knowledge from the user in order to be used, with the main disadvantages of being highly dependable on external and online modules in order to be successful.

2.5.4 OAQA

The other system from which MoQA has taken inspiration is from OAQA, which integrates additional biomedical and general-purpose NLP annotators, machine learning modules for search result scoring, collective answer reranking, and yes/no answer prediction, compared to its predecessor from the same author.

In this system, there is a wide use of external software, such as the TmTool [109], and MetaMap, in order to identify possible biomedical named entities. Also, as seen in 2.5.4, there is the extraction of frequent multi-word terms from relevant snippets to further improve the recall of concept and candidate answer text extraction. Also, much of the previous work is reused, such as BaseQA, which is designed for domain-independent QA components and includes the basic input/output definition of a QA pipeline. It is interesting to note that although this software has similar problematic characteristics as described from the other systems in general. For clearance, the TmTool provides a standard Web service interface to annotate biomedical concepts using a number of state-of-the-art biomedical NLP parsers.

As for answer type prediction, a number of linguistic and semantic features are extracted from the tokens and concepts, including the lemma (a heading indicating the subject or argument of a literary composition or annotation) form of each token, the semantic type of each concept in the question, among other metrics, where the concepts are identified for instance, from MetaMap. When it comes to candidate answer generation, depending on the question type the system applies different strategies to generate candidate answers. Finally, as for candidate answer scoring, the system defines a group of features to capture how likely each candidate answer is the true answer to the question from different aspects, which includes candidate answer occurrence count, name count, etc. A Logistic Regression classifier is used to learn the scoring function, where the class is weighted by their frequencies.

For concept identification, MetaMap is used along with the GeneTag corpus. Here the authors admit to some clutter from other external software used, which forces to additional measures [117].

When it comes to answering ranking, the method used aims to boost the low-ranked

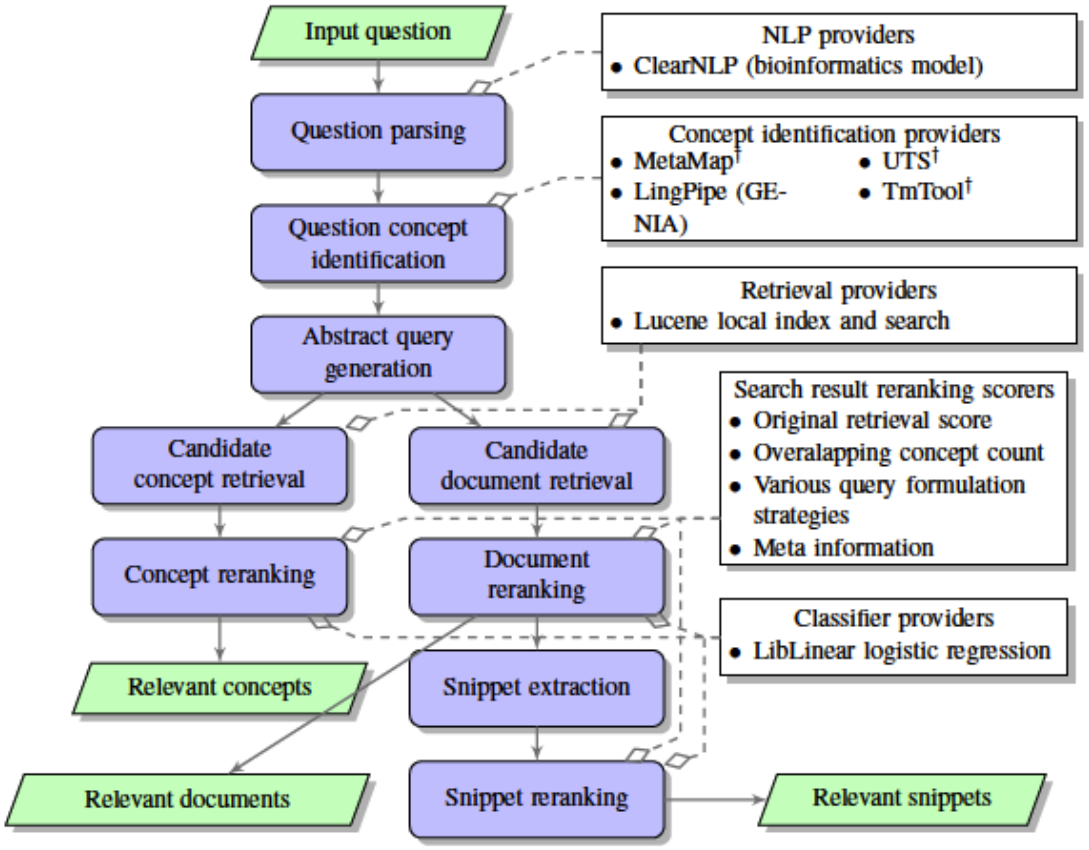

Figure 2.1: OAQA's pipeline diagram for BioASQ Phase B of 2015 challenge, with access external Web services. [117]

candidate answers, which share the same semantic type with high-ranked candidate answers for list questions. After the answer scoring step where a confidence score is assigned to each candidate answer individually

As for Yes/No question answering, the authors consider it to be a binary classification problem. An interesting idea comes to the surface for detecting bad answers, in which if a statement is wrong, then the relevant snippets should contain statements that are contradictory to the original statement, but the authors also admit to the difficulty of such task.

For the retrieval of relevant documents, concepts, and snippets, first it is retrieved a list of 100 candidate results, then a set of features is defined to estimate the relevance of each candidate result and employ a standardized interface to incorporate these features. In this stage, Lucene indexes are used contrary to GoPubMed [26] ones, and performance improvements are duly noted.

As for OAQA, it is a system designed with a biomedical use case, with a large quantity of documentation, pre-requisites and rather heavy configuration installation and configuration, which is only doable by knowledgeable individuals. Their philosophy included

not a modular systems that are improved from module to module, but rather the system as a whole.

2.5.5 Other Existing Systems

From the survey consulted [40], about 86% of the systems present in the survey showed to be highly different from on another, which show low reuse of the systems and their characteristics and/or implementations. For the Semantic Question Answering (SQA) systems, some methods such as natural language (NL) parsing and part-of-speech (POS) tagging, are available and commonly reused, but even so, in distinct ways.

In [38], it is proposed a SQA system which uses syntactic dependency trees of input questions, and this method consists of three main steps: **(1)** Triple patterns extraction using the dependency tree and POS tags of the questions; **(2)** entity, property and class extraction and mapping to the underlying knowledge base; **(3)** question words matching to their respective answer type. The best-ranked answer is then returned. PARALEX [28] only answers questions for subjects or objects of property-object or subject property pairs, respectively. QuASE [95] is a three-stage open domain approach based on web search and the Freebase knowledge base: **(1)** entity linking (determining the identity of entities mentioned in text), semantic feature construction and candidate ranking on the input question are used; **(2)** the documents and according sentences are selected **(3)** from a web search with a high probability to match the question, presenting them as answers to the user. In [95] web search engines are used to extract relevant text snippets, which are then linked to Freebase, where a ranking function is applied and the highest ranked entity is returned as the answer.

HAWK [103] is the first hybrid source SQA system which processes linked data as well as textual information to answer one input query. HAWK uses an eight-part pipeline made of part-of-speech tagging, entity annotation, dependency parsing, linguistic pruning heuristics, semantic annotation of properties and classes, generation of basic triple patterns, discarding queries containing not connected query graphs, finally ranking them.

OpenQA [62] is a recently introduced end-to-end QA pipeline platform, which shares the same goal of YodaQA of a common research platform in the field. However, in OpenQA there is more of a portfolio-style engine with mostly independent pipelines which have their candidate answers combined, while YodaQA emphasizes modularity on the pipeline stage level.

2.5.6 Other issues

In general, there are other problems present in the described systems. One of those problems is the lexical gap, which is found in specific domain systems, the Lexical Gap: the difference between the vocabulary used for a question and the one used for the labels.

PARALEX already tries do solve this issue by using the corpus of WikiAnswers. Another issue with these systems is that they generally are not updated or even backslide technologically.

Chapter 3

Developed Work

This section describes all the work developed spanning several months, in order to accomplish the proposed objectives. One of the main goals is the development of an easy-to-use, adaptable system with acceptable performance, capable of answering user questions given two sets of data: a set of documents in which answers will be searched, and training data, in order to train a document classifier.

First, an overview of the system is presented, where its architecture from a higher level of abstraction is described, followed by a detailed description of each of the system's modules and respective sub-modules, along with the datasets, tools, and datasets used by them. This system is a result of two previous iterations: the first version was developed by the team ULisboa, where I was one of two main developers, and it was used to participate in the BioASQ 2016 edition. This is the first system described in this section, along with its results.

The second version was developed for the SemEval 2017 edition, which took place during the course of this work (october 2014 to january 2015) and it is described in this chapter after the first one. These participations allowed me to assess the final iteration of the developed system, being its performance assessed as well in a closed environment.

3.1 WS4A

The first system, dubbed WS4A participated in the fourth edition of the BioASQ challenge of 2016 a challenge on large-scale biomedical semantic indexing and question answering.

WS4A was a joint venture developed along with another colleague of the time, Miguel Falé and it was used to take part in the Question and Answering (QA) task 4b [1], which consisted on the retrieval of relevant concepts, documents, snippets, RDF triples, exact answers and ideal answers for each given question from a JSON file by the organizers. This system focused on the maximum exploitation of existing web services in each step of WS4A, such as the annotation of text, and the retrieval of metadata for each annota-

[1] http://www.bioasq.org/participate/challenges

tion.The reason was because of the lack of time and knowledge of various implementations needed to be developed at the time. The information retrieved included concept identifiers, ontologies, and most importantly, PubMed identifiers.

Following is a description of the WS4A pipeline, and also after that the precision, recall and f-measure values obtained in task 4b. This system achieved second place twice in two subtasks on one of the five batches.

3.1.1 BioASQ 2016

The BioASQ competition of 2016 was composed of two phases: in the first one, the goal was to take a set of queries as input, responding with a set of the most relevant concepts, articles, snippets and RDF triples. The list of ontologies and terminologies from where concepts were to be retrieved include the Medical Subject Headings (MeSH), the Gene Ontology (GO), the Universal Protein Resource (UniProt), the Joint Chemical Dictionary (Jochem) and the Disease Ontology (DO). From those concepts, web scrapping was performed from the Linked Life Data project.

In the second phase, the organization provided gold responses to the same questions of the first phase, so this time, the goal was to search and provide a response given the correct documents and snippets provided by the organization and compiled by experts.

The questions were in four categories/types: Yes/no questions, that require only ”yes” or ”no” answers; Factoid questions, that require some kind of information or expression; List questions,which would be a list of factoids; Summary questions, that require a short text as an answer, such as a description. A general description of the system proposed by the organizers is present in Figure 3.1.1[100, 48].

In WS4A, the approach explored every possible option to use public web services and incorporate available domain knowledge. In WS4A, the above tasks were addressed by recognizing relevant terms in the query and also in the abstracts associated with it based on available Web Services. In the next step, the system developed mapped those terms to their respective concepts in ontologies and terminologies. Then, WS4A compared those concepts to identify the responses that had the highest frequency among shared concepts with the ones associated with the query. WS4A employed semantic similarity to measure how close in meaning they are even if they do no share the same concepts. Additionally, WS4A used Machine Learning [77] techniques to classify if an abstract is either relevant or not for the given query.

In this section, firstly Web Services explored by WS4A are described, and then the composing modules of the system are explained.

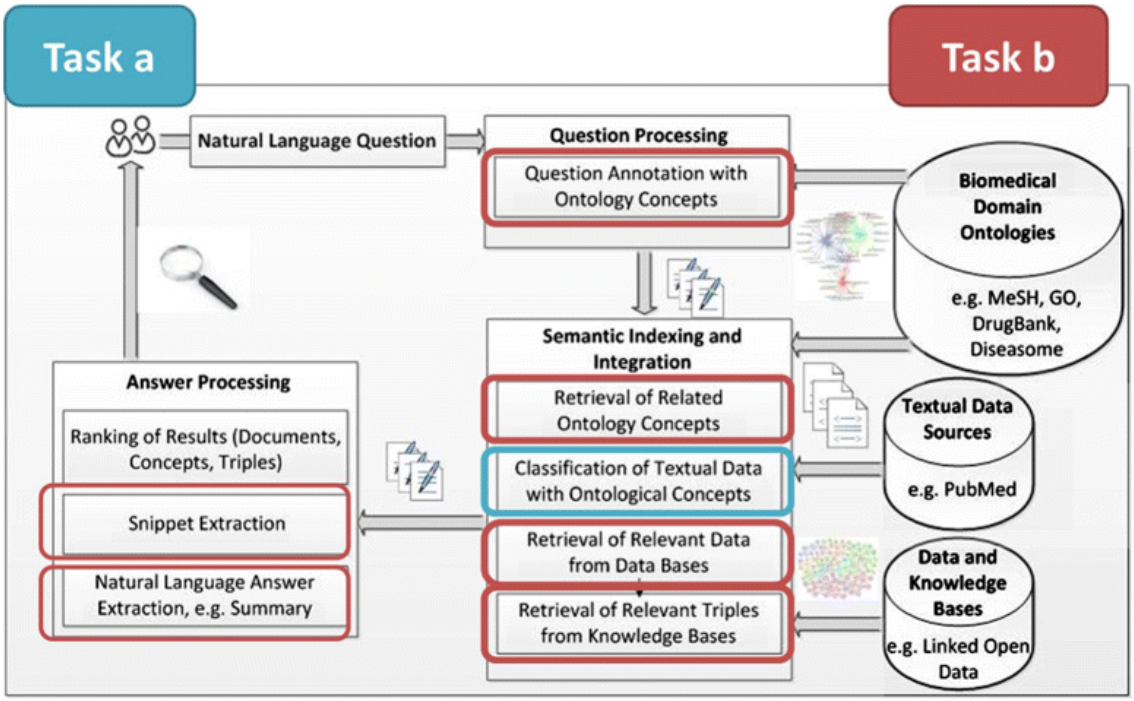

Figure 3.1: BioASQ organizers' overview of semantic indexing and question answering in the biological domain.[100]

3.1.2 Web Services

From the many Web Services used in WS4A, the first one is provided by BioPortal [73], where a given text returns the concepts from the ontologies required by the request mentioned in it. There are some other parameters that WS4A explored with the purpose of better filtering the results, such as i) longest annotation only; ii) number exclusion; iii) whole word only and iv) synonym exclusion.

With the UniProt Web Services [21], PubMedIds from protein descriptions would be gathered, with the resource of another Web Service, Whatizit[2], in order to obtain proteic descriptions. Whatizit [83] served as an alternative to BioPortal.

Using UniProt, the request is made in the following format: i) the Web Service URL; ii) followed by a protein identifier; iii) ending with the available chosen format. An example for "P12345" is present in Figure 3.1.2.

When it comes to retrieving PubMed identifiers (PubMed IDs), the PubChem Web Service [107] was the last one to be used. This Web Service provided an easy interface since it required trivial parameters and URLs. Each request would have: (1) a Base URL; (2) a concept in the question; and (3) a Data Format (i.e. JSON). An example of the request for the concept "oxygen" in the JSON data format can be seen in Figure 3.1.2.

NCBI has a service, eutils [88], in which two of them were used in order to retrieve PubMed articles by their identifiers, with the restriction that the articles had to be from

[2]http://www.ebi.ac.uk/webservices/whatizit/info.jsf

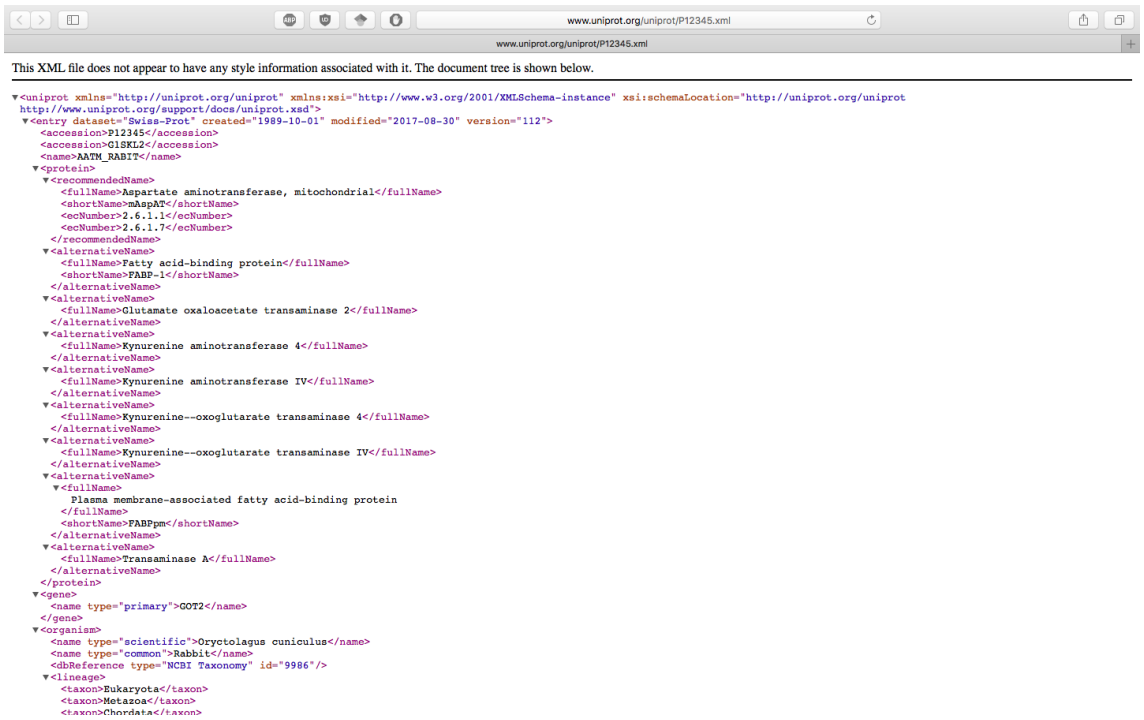

Figure 3.2: Use of the UniProt WebService.

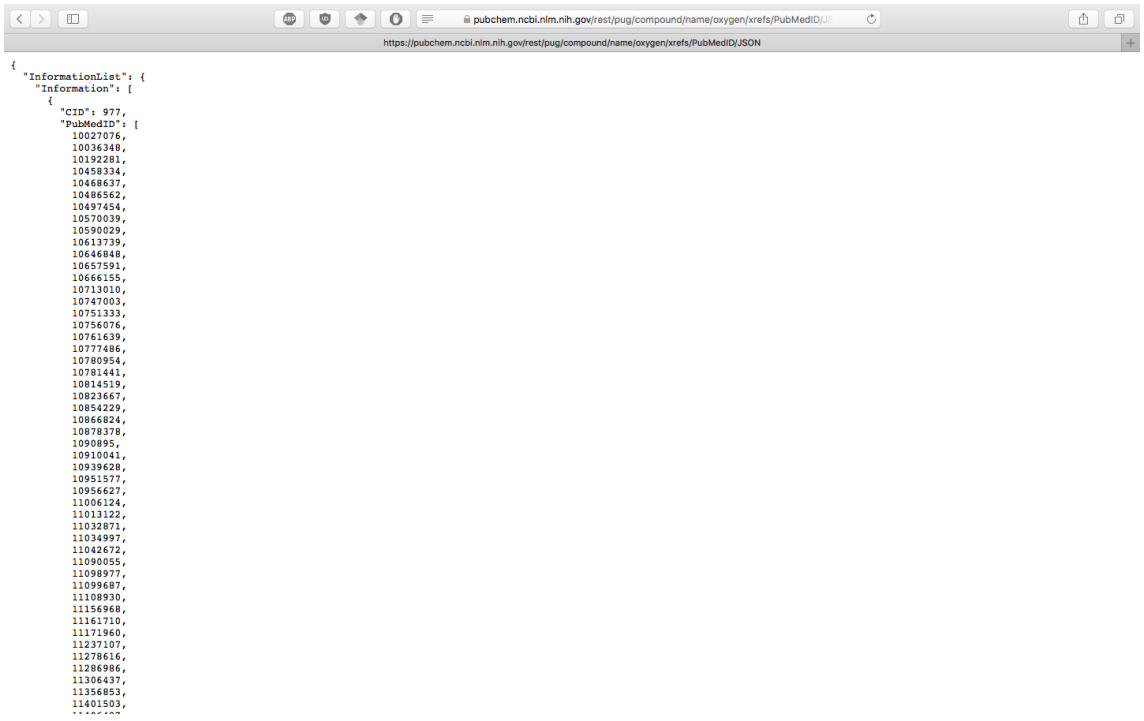

Figure 3.3: Use of the PuChem WebService.

before November 19th, 2015. The URLs are easy to identify and build since they have a specific format. An example, in Figure 3.1.2 where the article with the PubMed ID 223687640 is shown. From eutils two services were used, one that searches for PubMed

IDs with MeSH annotations (Figure 3.1.2); and one that fetches the abstracts from the Ids retrieved from the previous URL, as seen in Figure 3.1.2.

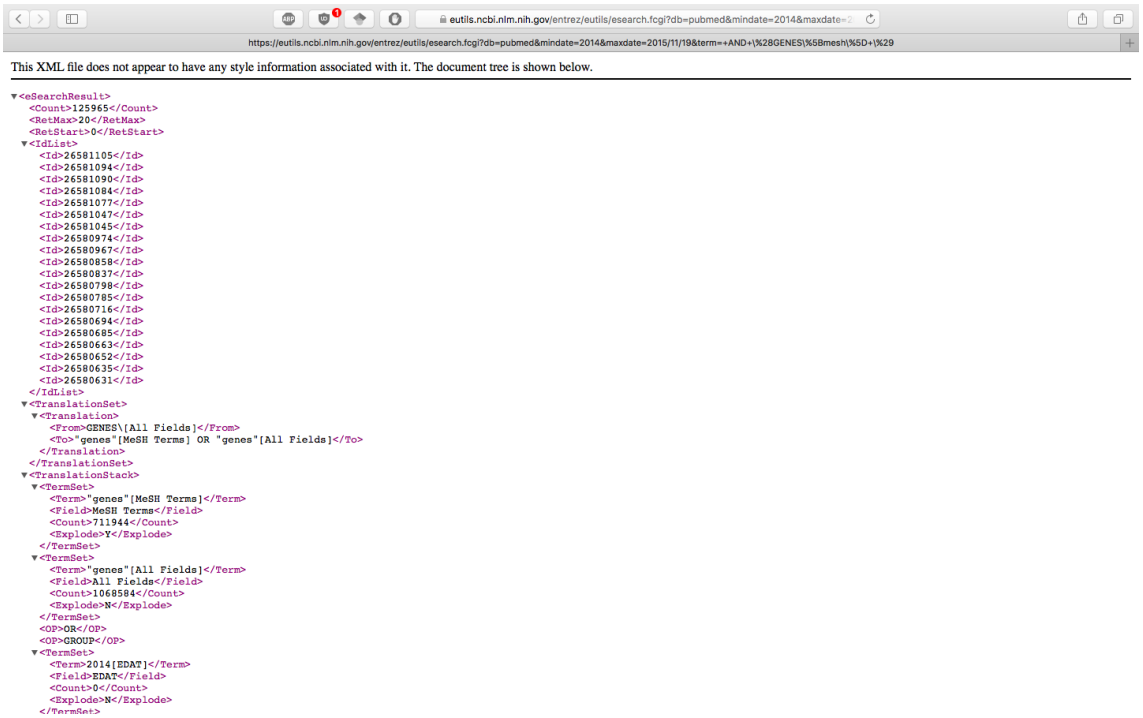

Figure 3.4: Use of the NCBI WebService.

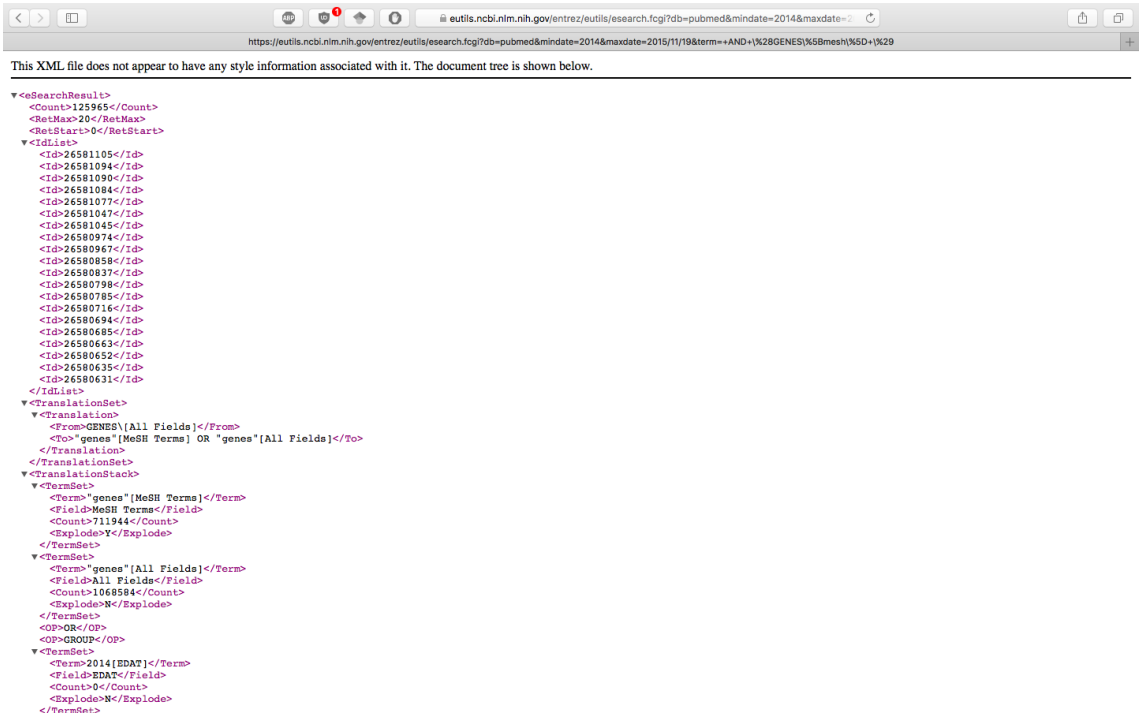

Figure 3.5: Use of the Eutils WebService for PubMed ID gathering.

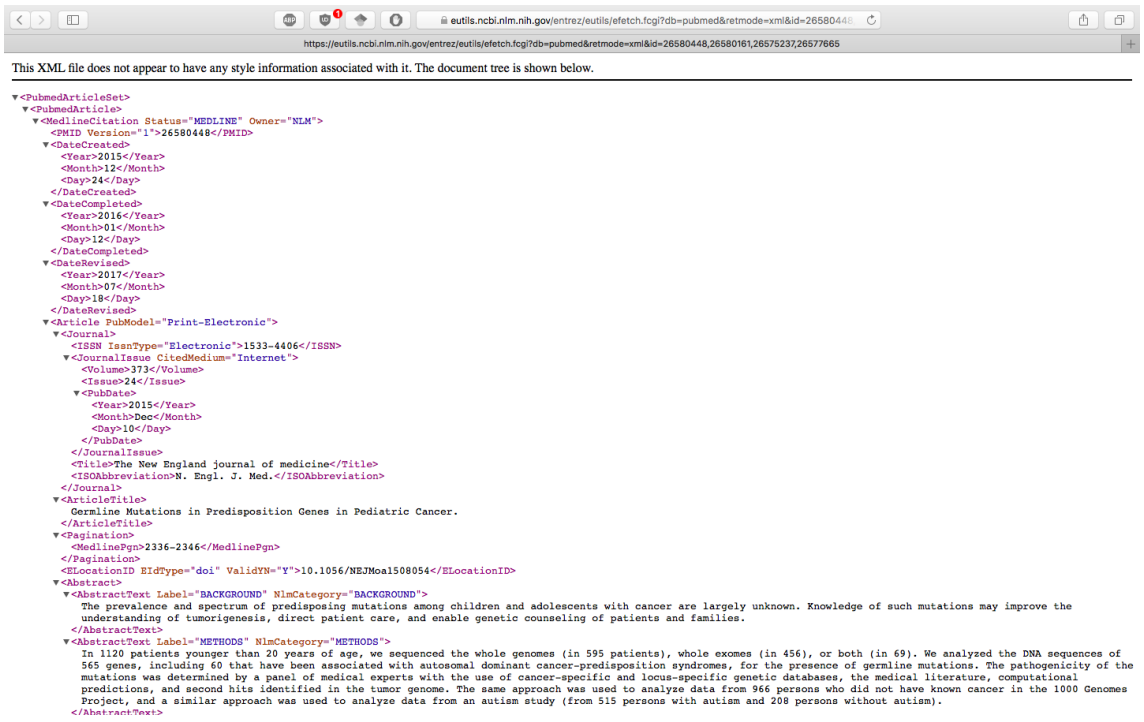

Figure 3.6: Use of the Eutils WebService for articles using specified PubMed IDs.

Finally, in the Answer Builder module, and another Web Service, from Linked Life Data and their SPARQL query endpoint, to generate RDF triples according to the ranked abstracts and annotations.[3])

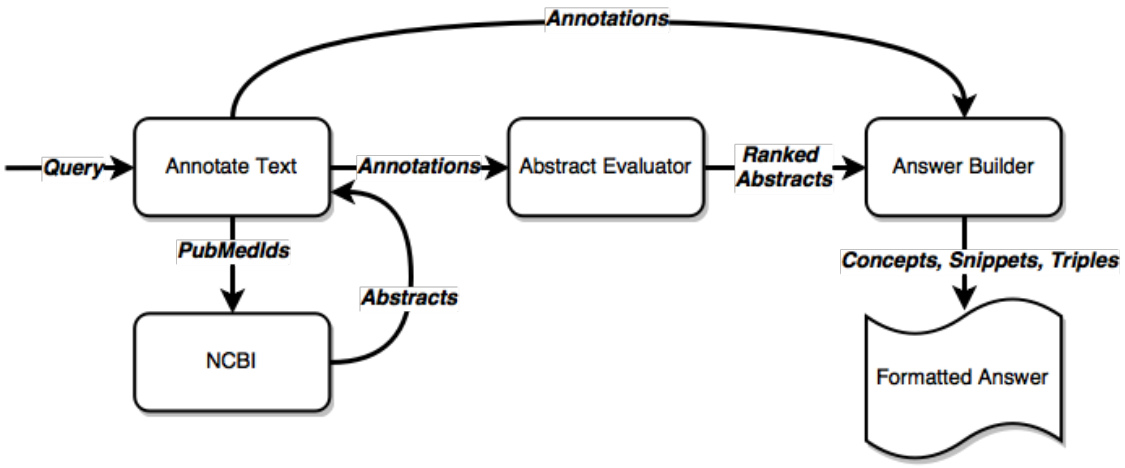

Figure 3.7: The pipeline of WS4A with its main modules.

[3] http://linkedlifedata.com

3.1.3 Pipeline

Figure 3.1.2 shows the modular nature of WS4A, where each module was designed to be as much independent from the others as possible.

The first module of this system, Annotate Text, is where question annotations by ontologies take place, as described in the previous section. After the PubMed IDs are gathered into one structure, only the ten most recent articles are used, independently of the ontology it comes from, just to improve temporal performance, because. Then, take these ten abstracts in order to obtain annotations from the abstracts per ontology, by running them through the Annotate Text module, in order to obtain annotations on the abstracts as well.

In the Abstract Evaluator module, only the abstracts that are deemed to contain useful information are used in the response. For each abstract, the following scores are registered for each abstract:

- Jaccard score between the query and abstract annotations
- Hierarchical distance score between the query and abstract annotations
- Frequency between the top semantic annotations of the abstract and query
- Sentence semantic similarity score

A grade then is given to the abstract using these scores. Depending on the operation being computed (training or classification), there are two possible courses of action: approval or disapproval of the abstract, so it can take part (or not) in the response in case of being in the training phase; otherwise, it adds information to the training arrays.

After the selection and ranking of abstracts, the Answer Builder module selects the 10 (according to the BioASQ's rules) best-valued snippets from the semantic analysis. Also, the concepts required for the response are generated. The concepts selected are the ones whose summed similarity score from all the abstracts belonging to the top ten.

The following step consists of taking these top 10 annotations from the abstracts which are MeSH annotations and then use them to generate the RDF triples. To filter the resulting great amount of RDF triples, TF/IDF comes into place in order to obtain those top 10 that add more content.

3.1.4 Data Training

Support vector machines [43, 77] were used to classify the relevance of the abstracts to a given query. From the Abstract Evaluation module, the selected working features were the four scores of the Abstract Evaluator module and the top 5000 n-grams (in frequency).

3.1.5 Features

The similarity features used include:

- n-grams (n = [1, . . . , 4]), after stopword removal (between texts for example)
- Longest Common Substring
- Jaccard coefficient
- Word containment (overlap, noun overlap), also known as tokenization
- TF/IDF
- Cosine similarity (question vs. documents)
- Word2Vec scores
- Longest Common Subsequence

3.1.6 Summary

This system showed the feasibility of developing a question and answering system mostly based on web services, in which other systems already had used, such as IIT.[106] WS4A also used WordNet, when comparing between the words in the query and the words in the abstracts, all based on web services and fully explores the semantics given by the ontologies. Thus WS4A is a light system that can be easily deployed, and which is continuously updated given the extensive use of web services.

3.1.7 Aftermath

As for future developments, it was concluded that improving the annotation gathering functionality was needed, including also other sources such as DBpedia[4]. YodaQA [78] resorts to DBpedia Spotlight, a service that automatically annotates DBPedia concepts from the plain text.

3.2 MoQA

From the work developed, a lot of research was made during the first few months, culminating in the overhaul of the WS4A system that took part in the BioASQ 2016 competition.

In order to materialize this system, the main and most advantageous ideas were taken from various works that in principle, as seen in Section 2, and as a result for MoQA, new

[4]http://dbpedia.org

modules were added and others revamped, which will be described, along with their justification.The systems that are in use nowadays are not usable for someone lay in the area of or even someone that has a knowledge base or a specific topic that would like to inquire about. So, the system proposed would be able to decouple the modules individually for use in other tasks, such as answer ranking, given a certain question, as seen in this section firstly in MoRS and then in MoQABio. This modularity serves so that anyone may tweak each module as one sees fit, or remove or even add modules if one has the will to.

Although the system developed includes both MoRS and MoQABio, since they had two different purposes and differed in input as well as output, they are described separately.

3.2.1 Approach

In this subsection, the approach in each area of MoQA as to its development is described. It goes through, preprocessing of data, annotation, chosen features for the classification module and answer presentation. Each external tool or resource should share a similar approach of use or even a black-box approach.

3.2.1.1 Preprocessing

The first step in building MoQA resides in the due treatment of the corpus for the task at hand. Since one of the objectives was to make it adaptable to many tasks, the preprocessing step is one where the users forcibly have to develop their own preprocessing scripts, and transform the data to a format in which after a few tweaks is readable by MoQA. This is because the files may be in practically any format (i.e. XML, JSON) and structure.

3.2.1.2 Annotation

The system is to include a general annotator, using Named-entity recognition (NER), with English as the selected language due to the quantity and quality of resources publicly available. This annotator would be applied in distinct domains by using narrative texts related to those domains. Although at first, it was thought of using an annotator with bidirectional Long short-term memory (LSTM, a recurrent neural network architecture) [33] with a sequential conditional random layer above it, along with a token-level evidence as orthographic evidence and distributional evidence, from [50]. Unfortunately, since I am not fluent with the use of C++, this was not the way to go, and since I had already worked with Stanford NER before, the second was the general annotator of choice, which works with Java. Stanford

3.2.1.3 Features

From a great number of features, the ones more related to the tasks at hand were selected. These features were somewhat different from MoRS to MoQABio.

Since machine learning will come to use further ahead, the decision on a set of features, and to distinguish when to use those same features, is an important decision. As for MoQA, there are three groups of features: to measure the similarity between questions and answers, to measure keyword density and frequency and finally those that measure the correlation between question-answer pairs. These features are useful to classify information such as the biomedical documents onto ontology concepts, classify biomedical questions on the same concepts, integrate important documents, snippets, and other information, such as in the tasks approached. [100, 48]

Text similarity features In text analysis applications, a common pipeline a usual pipeline adopted use similarity from concept level, to the word and sentence level. For example, word similarity is first computed based on similarity scores of WordNet concepts, and sentence similarity is computed by composing word similarity scores.

Some of these features were inspired by related work from each task, and also from WS4A.

There were other features available, such as the number of verbs, nouns, pronouns, and adjectives in the text being analyzed.

3.2.2 MoRS

The second step was MoRS (spelled 'Morse'), which participated in Task 3 of SemEval-2017. MoRS was used to perform the Community Question Answering Task 3, which consisted on reordering a set of comments according to their use when answering the question in its corresponding thread. This was made for a large collection of questions created by a user community from Qatar Living.

As described, the approach was to get a hold of simple, easy-to-use and laid aside technologies that, in the hands of non-expert people, could be reused in their own data sets. Some of the techniques included the annotation of text, the retrieval of meta-data per comment, POS tagging, and NER. After these, syntactical analysis and semantic measurements took place.

This was the first system were some ideas developed would be tested, by trying to use the system in cQA first. While QA systems rely on a user query in order to search and prepare an answer based on the searching capabilities it already has and its documents, in cQA the query and respective related answers are already provided, being only necessary a reordering by relevance of such answers, or perhaps even a rephrasing of such an answer in order to suit better the query.

What came about different from WS4A, was that this time in cQA, a user resorts to the web for answers without a given structured knowledge base, relying on the arbitrariness of cQA forums, the dependence and waiting time on their results, which may slow the gathering of answers in real time. Also, public forums are dependent on the users' input (i.e. answers), which might be rather unstructured, not straight to the point, not related to the question at hand, not well written (i.e. grammatically), lengthy or even incorrect. [25, 39, 105]

MoRS used Machine Learning [77] techniques to classify if a comment as "Good", as explained in the description of the Subtask, and "Not Good" according to the comment's relevance.

Despite successful implementation, the desired results were not achieved due to data set corruption found only after result submission.

3.2.2.1 Datasets

The data provided consisted on 6,398 questions with a total of 40,288 comments. Each comment in a given forum thread would be labeled as "Good", "PotentiallyUseful" or "Bad".

3.2.2.2 Preprocessing

When treating the corpus of questions and respective answers, the idea was to cut out those which might be of dubious nature, i.e. questions and answers that have at least four words each, with at least one noun and one verb. The idea is that good answers have a minimal amount of structure. The text would be split at the sentence level, tokenized and POS tagged. Each word would be morphologically simplified using the WordNet [66] library. Other steps include stopword removal, lemmatization and stemming. [31, 96]

3.2.2.3 MoRS Pipeline

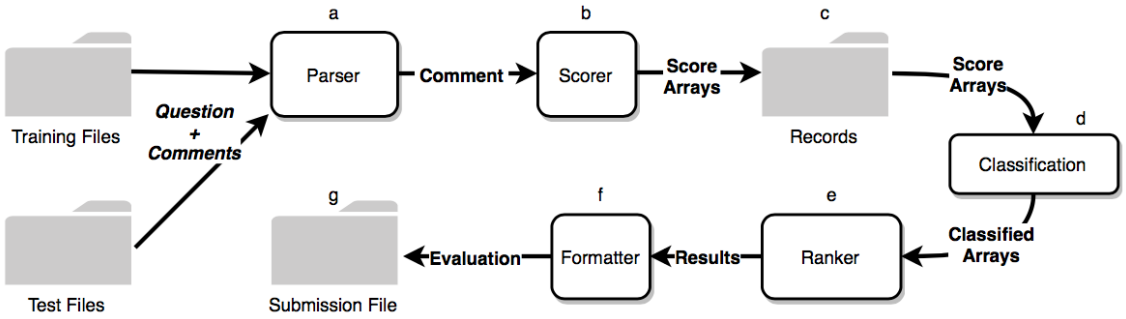

Figure 3.8: A schematic of the first version of MoQA in action, tailored for a ranking task. [85]

As seen in Figure 3.8, the system is separated and defined in several modules that work as a pipeline, where each module was designed to be as much independent from the others as possible. It is possible to see that, although adapted to the task at hand, much of the previous modules from WS4A are repurposed in MoRS: Annotate Text has become Parser (a), Abstract Evaluator has been divided into the Scorer (b), Classification (d) and Ranker (e) modules, and Answer Builder was refurbished into Formatter (f), therefore maintaining the same structure and inherent logic, but with better separation and more specific sub-tasks for each module. In the first step, XML files provided go through the XML parser *(a)*. This parser is tailored to the structure of the XML in the file. The information extracted from the parsing is the question, its author, and from each comment, the author, the text that composes the comment and, if in training phase, the golden score of the comment. Each comment then goes through the Scorer module *(b)*, going through various scoring methods which involve, **(1)** cross-matching of Named Entities (exact and partial tokens), using Stanford Named Entities Recognition [30], determining the number of named entities that the comment has in common with the question; **(2)** if the author of the comment is the same as the author of the question, giving it away that such a comment would not be fit, since the questioner only on rare occasions answers his own question; **(3)** if the comment has any question marks, proving that that comment does not answer the question by not being assertive; **(4)** if there is any swear words or even **(5)** misspelled words (from a given list), that may demonstrate a lack of zeal in the answer, plain ignorance or at least lack of effort from the author when answering; **(6)** sentence semantic similarity score for sentences between the question and the comment, based on the Wu-Palmer metric [113]; **(7)** if there are any personal pronouns, making the notion of opinion making, which may indicate an answer to the question; **(8)** the presence of other question URLs or even **(9)** image URLs which might indicate that a question might be already answered in another thread; **(10)** the existence of nouns in common may point to similar concepts in discussion in the comment; **(11)** the presence of smileys, from related work showed that they are not a good sign or assurance in this way, because they did not show seriousness from the comment's author; **(12)** the number of comments from that author, in which a relatively large number indicates that the author has many comments in that thread that at least do not answer correctly the question, therefore the need of other comments. Finally,**(13)** the length of the comment and it's ratio **(14)** with the length of the question.

The resulting arrays are used for *(c)*, the classification phase *(d)*: with all questions and respective answers are dealt with, SVM is used to classify between "Good" comments and other comments, based on the information provided in the training files.

The "training" phase from the classification ends here, in step *(d)*.

In the second phase, the test files go through the same modules as the training files did, with the only difference being that in the classification module, the resulting arrays are

classified as "Good" or not, then going through an implementation of SVM^{light}, SVMrank *(e)*.

SVMrank has a learning phase, where the scored arrays from the training files were provided.

Formatter After the ranking scores for each question is given, these are run through the Formatter module *(f)*, where the submission file is prepared according to the Subtask's requirements specified in the instructions *(g)*.

3.2.3 MoQABio

MoQABio is the final and culminating of all this work. It has the suffix 'Bio' since it has in its annotator module a NER for biomedical text. This system was purposed to be tested with data from the same challenge WS4A took part, BioASQ 2016, in order to compare results and test if the system performed better while maintaining the framework of MoQA. The exact same data sets were used, those provided by the organization of 2016. In difference to MoRS, some modules were added, other were upgraded or modified, and others were not used since the task had a different purpose. The approach used on MoRS was the same used on MoQABio, which included accessible tools, techniques, formats and textual measurements, some alterations among the modules to better fit the data in question, and the conjunction of the input and output of the modules.

As the last and most complex system developed, contrary to MoRS, MoQABio was to focus on being a Question Answering system, which in the end provided an answer according to the type of question selected, accompanied with support and metadata. This system relies on the documents made available to the system and its analysis of such documents as a contribution to the answer.

In relation to WS4A, MoQABio was not to resort to resources that were not local, but as the system's description will show, there were exceptions that were necessary in order to improve the system.

MoQABio also used Machine Learning techniques described in section 3.2.3.3, but this time to classify biomedical articles as useful or not as to answer the question.

The results will show the constant successful improvements of the system, and leave great hopes for future work and projects.

3.2.3.1 Datasets

Two types of datasets were used in this system.

Training The training data is the same as the one used in WS4A, which consisted of a JSON file with 1307 questions. Each question has a question, an exact answer (or a list,

depending on the type of answer), an ideal answer (depending on the type of question (or a list, depending on the type of answer)), snippets (from the list of documents), concepts, (RDF) triples, and more importantly, the documents (from PubMed) used in the answer.

PubMed Since the objective was to have the least Web Services in order to make the system more responsive, a subset of documents from the whole database, which added to 140Gb. The datasets were submitted as documents into Elasticsearch with similar mappings (in JSON).

3.2.3.2 Preprocessing

Since it is not wise to have a system that occupies such a large amount of memory, the PubMed articles were filtered if they were from after 2007. The other requirements when choosing PubMed articles were the existence of an abstract, title, for query search among Elastic search, with MeSH headings and chemical substances for NER comparison. These were also used for scoring. Each question would also be tokenized and POS-tagged, with the tagger of the users' choosing. Other steps include stopword removal and stemming.

3.2.3.3 MoQABio Pipeline

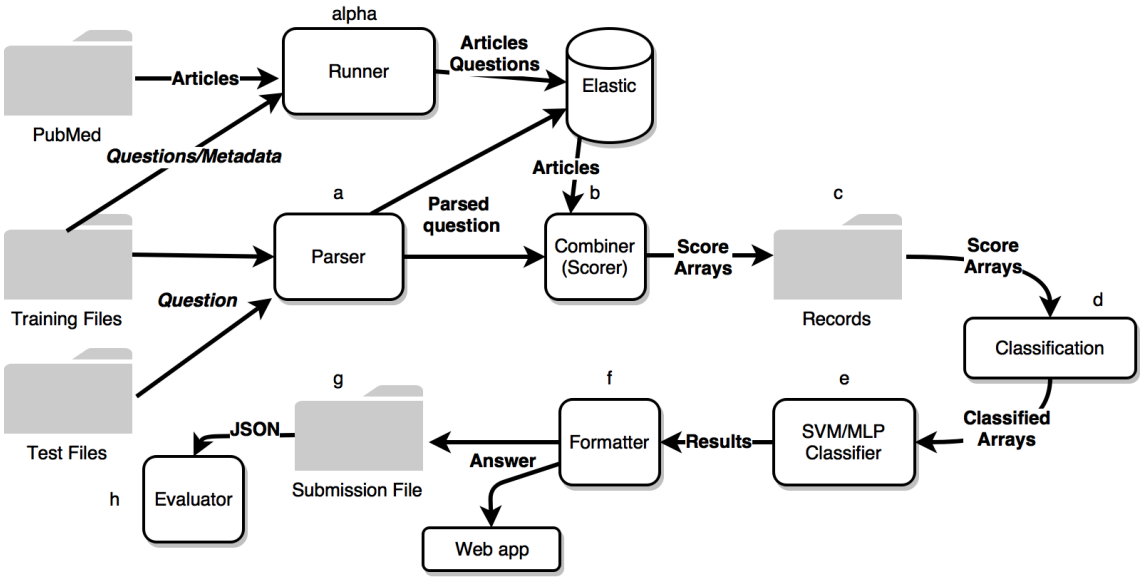

Figure 3.9: The pipeline of MoQA with its main modules, this time built for a biomedical question answering purpose.

In comparison to the MoRS pipeline in Figure 3.8, one may see a similar structure and modules compared this time to Figure 3.9, along with new modules, hence the modularity of the system. If one even overlaps these two structures, much will even coincide, starting with the equally named modules such as the Parser (a), Classification (d) and Formatter

(f), followed by the Scorer module, this time renamed Combiner (b). As for WS4A, its NCBI module has turned into queries to the Elastic document base. The following paragraphs describe in more detail each module in question, providing also with some details of the specificity of implementation.

Runner The runner is designed as the **alpha** step to the system, where all pre-configurations of the system are made, beginning with Elasticsearch mapping initialization. Elasticsearch mappings are the definition of the structure of the documents Elastic will index and search over in other stages. These mappings include the return of OpenPHACTS, DBpedia, PubMed articles and answers from the challenge in question (BioASQ). Here is an example of such mapping, specifically the mapping of the return of an OpenPHACT request:

```
{'mappings':{
    'phact':{
      'properties': {
        'queryText':{'type':'text'},
        'similarityScore': {'type':'float'},
        'surfaceForm': {'type':'keyword'},
        'support':{'type':'long'},
        'URI': {'type':'text'},
        'percentageOfSecondRank': {'type':'text'}
      }
    }
  }
}
```

Besides all the mappings for the other enumerated structures, the articles and questions are added to Elasticsearch for further search (of articles) or addition to the question pool.

Finally, the Stanford NER and Genia Tagger are initialized, for performance purposes, so they are only initialized once, and not every time they are called to parse/tokenize, along with the Sematch module, where some of its libraries require initialization as well.

Process In order to build such system, it had to go through a process, which began with the reading of the training file in JSON, which contained questions in various formats, themes, types of question, etc (**alpha**). Each question is dealt at a time and follows immediately to the text parser **(a)**, where it goes through the Genia Tagger, acquiring all the tokens that are nouns (that include 'NN' in their designation), and if none are found, it goes through the Stanford NER, acquiring all the Named Entities (NE) that are not designated as 'O'.

The next step involves taking the question, and query Elasticsearch for the set of top 10 articles it returns, ordered through its own scoring algorithm, described in Section 2. This number was selected due to it being the maximum reasonable number of articles

that could be analyzed in order to give a timely answer of around one minute, while maintaining performance. The query itself refers to substances and MeSH terms that are coincidental or close to the parsed terms of the question, and the similarity between the question itself to the abstract and the title of the article. The Elasticsearch's ranking algorithm, which is dependent on Lucene's Pratical Scoring Function score, replaces the ranking module existing in MoRS, where the most relevant pieces would be ordered. This way, a faster way of gathering documents is achieved, while filtering most of the articles. Of note, every request made to Elasticsearch was a REST request done through a tailored library provided by the Elasticsearch creators.

Features With the parsed question and through each article (which is also parsed through the same parser used in the question) and respective metadata, the system then goes through what is called the Combiner **(b)**, were the following set of features is used:

- Number of entities in common between the question and the given abstract
- Number of entities in common between the question and the list of MeSH terms of the article
- Number of entities in common between the question and the list of Substances of the article
- Seven averaged scores of the Word Similarity metrics between entities in the question and the article 2.2.2.1
- Seven averaged scores of the Concept Similarity metrics between entities in the question and the article 2.2.2.1
- Two averaged scores of the Entity Similarity metrics between entities in the question and the article 2.2.2.1
- Article length
- Short sentence similarity score
- Timestamp of the article

As the number of features coming from Sematch (16), one may conclude that the most was taken of Sematch capabilities, since the results they have had. Other features such as the ones that involved the cross-matching of named entities, ensued that higher the frequency, higher the probability of being about the same subject, or at least similar subjects.

Other processes Also, all these features sum up to 21, varying in type of feature, but in this module other processes take place, such as the continuous addition of Open PHACTS, DBpedia and SPARQL entries, in order to enrich the data to be provided in the responses and most importantly, avoid performance issues by constantly making repeated HTTP requests that hurt the performance of such a system.

Training With each resulting array, they are combined into the same file **(c)**, where the final training module, the classification module **(d)** comes into place. In this module, all questions and respective answers (along with their metadata) are taken into consideration, and SVM is used to classify between a useful article for that answer, and an article to be discarded in regards to that answer. Multi-Layer Perceptron is another option provided but only used for performance comparison purposes against SVM.

Testing In the testing phase, the test files (this time with only the question and type of question as the only provided data taken into consideration), go through the same modules as the training files did, but this time each article is classified as useful or not to the question at hand in the classifier **(e)**.

Formatter Since for the BioASQ task some of the same metadata is to be provided as in the training files (see the tasks description at [100, 48], and here is the time where the Formatter module **(f)** comes in. That same metadata (i.e. documents, snippets, concepts) is prepped for presentation in the final document for evaluation, but they also go through some evaluation themselves, although simpler.

Some of that evaluation is only done with data coming from approved articles by the classifier, where the selected snippets are from the top 10 scores using the small text similarity score, while the concepts come from the top 10 arithmetic total using the Word Similarity measurements. The triples are from the same concepts, until a 10 list of triples is completed, or all the concepts are gone through.

Finally, depending on the type of answer, the exact and ideal answer are provided in the required format. The submission file is then prepared according to the Subtask's requirements specified in the instructions **(g)**.

Evaluation Module An evaluation module **(h)** is provided and put into the last part of the MoQABio pipeline, giving Precision, Recall and F-measure scores for the test data set provided.

3.2.4 MoQA Web App

As one may see in Figure 3.9, there is a small module that is added to the final part of the MoQABio pipeline, which is a developed Web Application (Figure 3.10), where a user

may interact with the results of the system, as it would be in its testing phase, since it is the result of everything done before, as seen in Figure 3.11. In addition, a user even post new questions (with the type of answer question pretended), as seen in Figure 3.12 receiving in return all the data of that answer, just like in the testing phase, but with only one question at a time. The user may consult all the data used for the answer in an organized fashion.

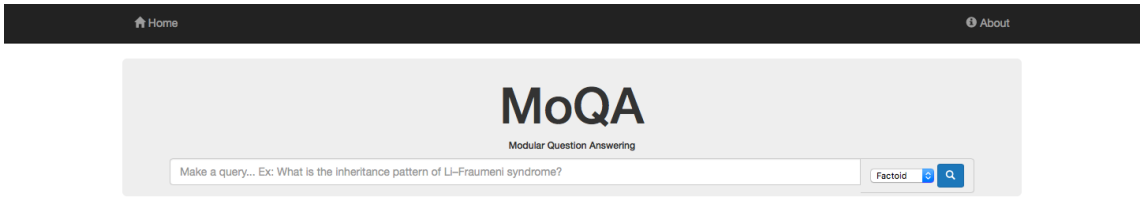

Figure 3.10: The home page of the web application developed for user interaction.

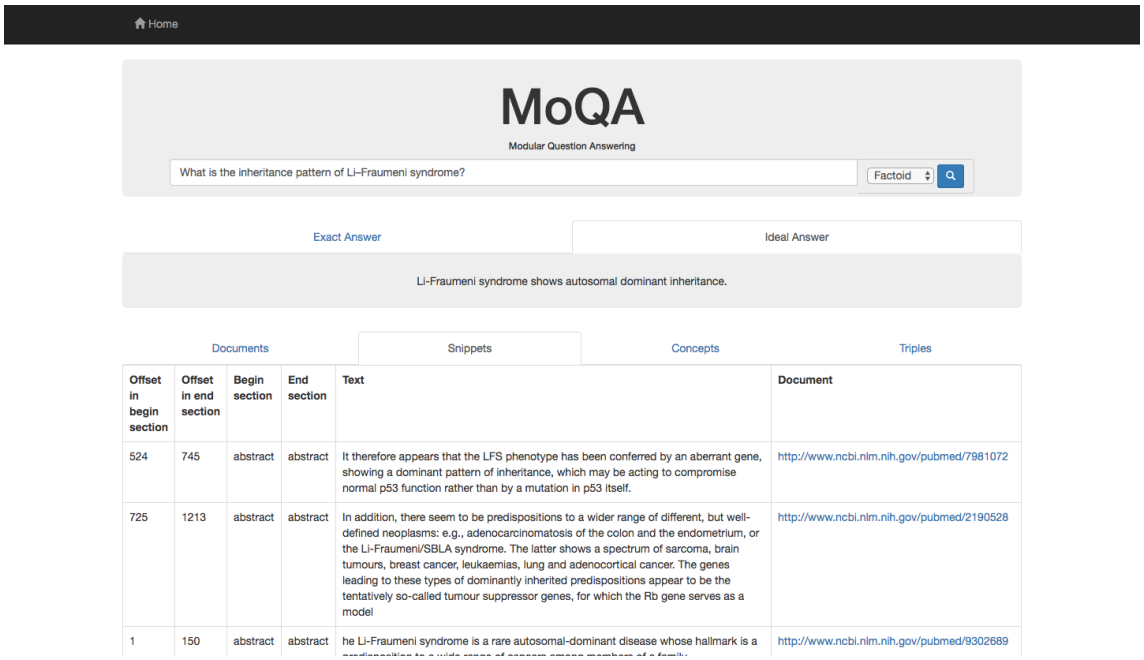

Figure 3.11: The answer to the faction question 'What is the inheritance pattern of Li–Fraumeni syndrome?', with the Ideal Answer and Snippet section opened.

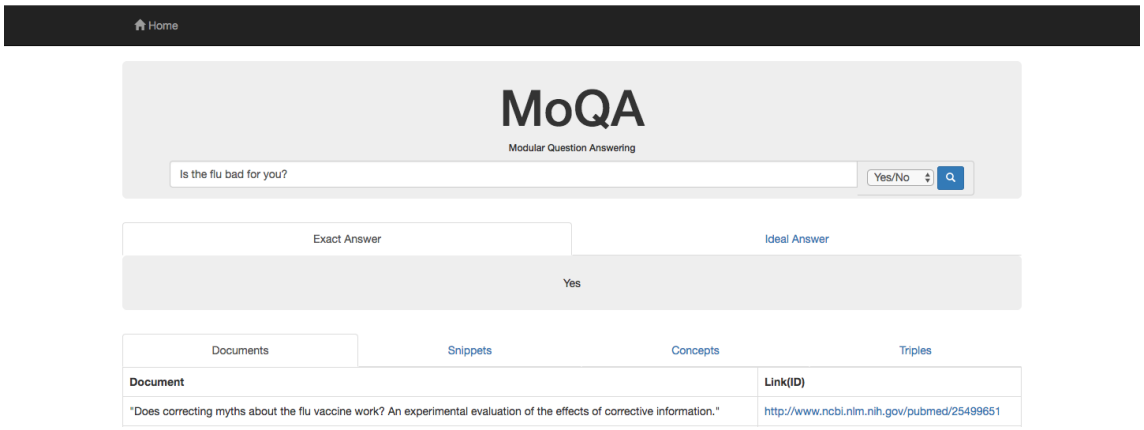

Figure 3.12: The answer to the user posed Yes/No question 'Is the flu bad for you?', with the Exact Answer and Document section opened.

Chapter 4

Results

This chapter, the results for each challenge will be commented, but not before the system is summarily described. As a whole, official and non-official examinations were made, and the two kinds will both be presented. The official evaluations are from the challenges in which the various versions of the system took part, in a chronological fashion, while the unofficial evaluations took place in a closed set, locally.

Not only of scoring evaluations were made, but of elapsed time as well. All metrics are rounded to three decimal cases.

In the end, a discussion about the overall improvement of system step-by-step is provided, with a culminating comment.

4.1 Evaluation Data Sources

The BioASQ 2016 and SemEval 2017 evaluations use the data sources described in sections 3.1.4 and 3.2.3.1 respectively, with the BioASQ 2016 data sources were also used to perform local evaluations.

4.2 Assessment

The systems used to participate in BioASQ 2016 and SemEval 2017 were officially evaluated according to the official metrics and corpora described in sections 3.1.4 and 3.2.3.1 respectively. The non-official evaluations were performed using precision, recall, and F-measure, using the BioASQ 2016 performance assessment.

4.3 WS4A Results

Following, are some aspects and characteristics that involved the results of WS4A and some interpretative assertions towards results themselves. They will show the success of the idea, but with poor results as for the performance of the system itself.

		System		
		WS4A	Lowest Off. Place	1st Place
Documents	P	0.010	0.005	0.169
	R	0.016	0.006	0.533
	F1	0.012	0.005	0.228
Snippets	P	0.004	0.043	0.082
	R	0.006	0.110	0.171
	F1	0.005	0.046	0.092

Table 4.1: WS4A results for all batches using the final version.

4.3.1 Hardware

Table 4.1 shows WS4A results for every batch and using the final version of the system. The times were obtained in a desktop computer equipped with an Intel(R) Core(TM)2 Duo CPU and 6 GB RAM.

4.3.2 Discussion

The results in Table 4.1 are presented with in comparison to the best and worst placed available official results published [48]. In this case only for the first batch and only for snippets and documents. It is possible to see the rather underachieving results of WS4A, but still proving the concept it intended to prove, while keeping its modularity. Of note is also the difficulty of such a task, by looking into the low scores of the best placed system.

The results as a whole, were expectedly among the lower half, since the lack of experience, time and knowledge at the time.

Although not present in the official results, in phase B, two second places were achieved for the first batch for exact and ideal answers, earning a prize from the organizers. The rest of the batches obtained the expected results stated before.

One aspect that made it difficult of improving the results from batch to batch was the time period given between them, along with other academic responsibilities. Also, in this year's competition, the type of answer wanted would already be given, so last year's results are not strictly comparable comparing to ours.

As a first experiment, WS4A proved to be a good proof-of-concept, by achieving good results in the first batch, while depending mostly on web services.

4.4 MoQA Results

This section reflects the results of MoQA, firstly with the test results of MoRS, and after those, the results obtained from MoQABio's evaluation. Beforehand, a description of the conditions of those tests and evaluations is made, along with their justification.

Submission	MAP	AvgRec	MRR	P	R	F1	Acc
KeLP	88.43	93.79	92.82	87.30	58.24	69.87	73.89
MoRS	63.32	71.67	71.99	59.23	5.06	9.32	48.84
Baseline	62.30	70.56	68.74	53.15	75.97	62.54	52.70

Table 4.2: Results from the Test Set of 2017.

Submission	MAP	AvgRec	MRR	P	R	F1	Acc
KeLP	79.19	88.82	86.42	76.96	55.30	64.36	75.11
MoRS	***81.15***	***81.42***	***88.44***	***74.37***	***99.94***	***85.28***	***74.34***
Baseline	45.56	65.42	53.50	34.44	76.41	47.47	43.32

Table 4.3: MoRS' comparison to last year's task 3 results.

4.4.1 Hardware

. Taking into account every kind of user, all the development was made on a local and regular computer, so that the computational requirements could be most accessible. Specifically, this tests were made on a MacBook Air (13-inch, Mid 2013), with 1,7 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 RAM memory.

4.4.2 MoRS Official Results

The results here presented in this section will go through a comparison with other teams' results and an explanation of why although apparently bad results, the system has proven itself.

This discussion will be tripartite, following the developments of the official results, followed by a comparison of the scores of the development set of the same year, concluding with the previous challenge test set results.

4.4.2.1 Discussion

The results placed us on the bottom of the table, with the best MAP result belonging to the KeLP team of 88.43, the result of 63.32, and the baseline just slightly lower of 62.30.

As one may see in Table 4.2, the results were quite similar to the baseline approach, which was close to random. To our surprise, MoRS was far from achieving a comparable performance, but after verifying the classification module which classified "Good" from not "Good" answers, it was noticed that the module was deficient, lacking about 95% of the arrays necessary to build it, due to a small error in the pipeline, which did not come about as a warning of any kind, and continued regardless.

4.4.2.2 Discussion after Re-run

After re-running MoRS, and making a flow analysis of the system, and improving some minor areas, the results from MoRS using the datasets from last years' task (Table 4.3)

Submission	MAP	AvgRec	MRR	P	R	F1	Acc
Beihang	0.714	89.2	77.265	-	-	-	-
MoRS	***79.91***	***80.02***	***86.51***	***74.39***	***100***	***85.31***	***74.39***
Baseline	45.56	65.4	53.50	-	-	-	-

Table 4.4: MoRS' results for the development set of 2017.

and the development set of 2017 (Table 4.4), confirmed that in fact, this year's results would have been much better if the models were correctly built.

These results were attained by using the organization's own datasets and scoring algorithms provided on the competition's page.[1]

Development set Discussion As one may notice, the development results, from the organization's scorer, show that in fact the problem was with the model, achieving better results at the time than any other submission. Given that not everyone published results for this submission, it is still clear that the error was part of the past, with large improvements all around in every metric.

Of notice, the maximum score of 100 in recall, meaning that all relevant instances were recognized, along with the first place way ahead of second place team Beihang.

One more test was missing, the test results from the test set of last year, since they were the most recent test scores publicly available.

2016 Test set result Discussion Coinciding with what was expectable from the previous table of results, the scores in Table 4.3 were quite similar to what to those belonging to the development phase, consisting of a MAP score of 79.91. Highlight to a maximum score of almost 100 in the Recall, a recurrence as it seems.

Another thing noticed, is that SVMrank had a very similar behavior in both set of results, so one may conclude that the issue of the results was exactly the classification of good answers, which also brought the results to a surprising first place if it had participated in the SemEval 2016 Task3. This was mainly because of the research of features used by the teams in that year's task and choosing what was thought to be best fit the purpose of this task. The scores for both 2016 test set and 2017 dev set are available in `https://github.com/migueljrodrigues/MoRS-Scores`.

4.4.3 MoQABio Results

In this section a similar assessment will be made, comparing this time the results of BioASQ 2016 with the performance results acquired by MoQABio. The focus of such assessment will be given to the gathering of information regarding the questions of the

[1] `http://alt.qcri.org/semeval2017/task3/`

		MoQABio	
		SVM	MLP
Documents	P	0.159	0.144
	R	0.317	0.313
	F1	0.191	0.167
Snippets	P	0.085	0.082
	R	0.144	0.139
	F1	0.095	0.101

Table 4.5: MoQABio's results regarding the BioASQ 2016 test set for snippets and documents, with a direct comparison between a SVM and MLP classifier.

test file, which means the documents and snippets used in that answer since these are the most important pieces of information to it. In addition to that first comparison, some elapsed time elations will be between MoQABio and the first system WS4A. Of note, the very same files used to train and test WS4A were the same used in MoQABio's training and testing.

4.4.3.1 MoQABio in BioASQ 2016

Documents assessment As expected, the results shown in Table 4.5 are quite on par with the competition's top, placing among the top 3 in precision. The recall might not be as high as other systems due to the shortage of documents it had access to since the PubMed repository has much more documents (around 140Gb) than the ones filtered into MoQABio (around 6Gb). The score of the F-measure would also be affected by the lack of a better recall score. Leveling the quantity of local documents might be a hard task, since the lack of physical memory might bet to be a problem, but there is always the last of using a web service publicly available to search for articles, with the cost of performance in both scoring and time.

Snippets assessment The quality of the snippets is totally dependent on the documents gathered since all snippets come from every selected and approved (by the classifier) document.

The results shown are quite encouraging, with a top result of Snippet precision and f-measure and a second place with the recall score.

SVM vs MLP As for the comparison between the two classification techniques, both scored quite similarly, with he expected edge going to SVMs, which often outscore in this kind of task. This happens since the dataset is still not large enough (i.e. no. of training set above 10000), or even because of the dual nature (good article vs bad article) of the build classifiers benefit more of the SVM's characteristics.

		System			
		MoQABio	MoQA	Lowest Off. Place	1st Place
Documents	P	0.159	0.103	0.005	0.169
	R	0.317	0.181	0.006	0.533
	F1	0.191	0.101	0.005	0.228
Snippets	P	0.085	0.038	0.043	0.082
	R	0.144	0.030	0.110	0.171
	F1	0.095	0.048	0.046	0.092

Table 4.6: In addition to MoQABio, another run was made without the biomedical parser and without resource to any other biomedical tool or library.

		MoQABio	MoQA	WS4A
Documents	P	0.159	0.103	0.006
	R	0.317	0.181	0.007
	F1	0.191	0.101	0.005
Snippets	P	0.085	0.038	0.002
	R	0.144	0.030	0.002
	F1	0.095	0.048	0.002

Table 4.7: Three-way comparison between the three developed systems, with MoQABio winning by a landslide.

4.4.3.2 MoQABio vs MoQA

Impact of biomedical components In this second testing demonstrated in Table 4.6, it is made quite clear that the technologies used in MoQABio, such as the Genia parser, the use of Open PHACTS, that separate it from MoQA, have an impact of the general results acquired. For instance, the annotated text from Stanford NER against Genia turns out to produce quite different search parameters for Elasticsearch, and therefore, ensuing a whole different result of that system, including the combiner module.

On a statistical level, only the f-measure of the snippets in MLP was superior to SVM's snippet f-measure, with an average value of 5%, although meaningless, since the large disadvantage in every other metric of 7% in SVM and 4% in all other 5 metrics. As for the documents only, this difference corresponds to 9% for SVM and 5% in MLP.

A system-wise comparison will be made in the next section.

4.4.3.3 MoQABio vs MoQA vs WS4A

Discussion The results in Table 4.7 results show that MoQABio largely exceeds the performance of the other systems, proving the framework to be successful, since WS4A begins the modular framework idea, MoQA improves it by using better generalized modules, finalizing with MoQABio that shows that the biomedical tweaks and transformations made to the system showed even better results. Therefore proving that MoQA is a framework that provides good, varied and easy-to-use tools for QA.

		WS4A(1)	MoQA(2)	MoQABio(3)	(1)vs(2)	(2)Vs(3)	(1)vs(3)
Docs.	P	0,006	0,103	0,159	1617%	54%	2550%
	R	0,007	0,181	0,317	2486%	75%	4429%
	F1	0,005	0,101	0,191	1920%	89%	3720%
Snips.	P	0,002	0,038	0,085	1800%	124%	4150%
	R	0,002	0,030	0,144	1400%	380%	7100%
	F1	0,002	0,048	0,095	2300%	98%	4650%

Table 4.8: Final head-to-head comparison between the three system.

But how much of a difference did it make?

Well, one may analyze the results from Table 4.8 by stating that the greatest improvement in absolute scoring value was made through the implementation of MoQABio, with (rounded up) improvements of 2550% in precision, 4429% in recall and 3729% in f-measure in the document scores. As for snippets, the difference in performance consists of 4150% in precision, 7100% in recall and 4650% in f-measure.

MoQA, as a general QA system did also pretty well, with improvements of 1617% in precision, 2486% in recall and 1920% in f-measure in the document scores and 1800% in precision, 1400% in recall and 2300% in f-measure in the snippet scores. All this comparing to WS4A.

The specified system in domain knowledge, MoQABio,had improvements againt MoQA of 54% in precision, 75% in recall and 89% in f-measure in the document scores and 124% in precision, 380% in recall and 98% in f-measure in the snippet scores.

These results show great room for improvement, since the growth in performance is still steep.

Elapsed Time Discussion For WS4A, the average time to answer a question was of around 90 seconds, with slow internet connection sometimes delaying the answer procurement. Some of these results may even been affected by occasional and impromptu problems, such as query limits, and servers being unavailable. The elapsed time in MoQABio, for each question is about 75 seconds, shorter, although still far away from a real time answer time, but more useful and effective computations are made in MoQABio than in WS4A.

4.4.3.4 User Tests

MoQA's installation and configuration was tested in the three most popular operating systems, achieving different results among them, but with similar user experience. The framework was tested in MacOS, Windows 10 and Ubuntu 14. Although not tested in other operating systems, it is still possible that they work just fine, given that the prerequisites are fulfilled. For each of the three users, already experienced in computer science and engineering, the instructions given at `https://github.com/lasigeBioTM/`

MoQA were provided, being followed by the instructor if any problem should happen or any help is provided. The users had little to no difficulty in following the steps and coming up to a running system, with small exceptions described later in this section. The main obstacles found in the configuration and installation process were:

- the lack of permissions to run scripts of SVMRank
- the need to change the SVMRank default files to the target operating system
- the package manager for the MLPClassifier delivering such package with an error that requires manual alteration of such library
- the requirement of installation of specific versions of some packages so the system may run without any problems

Of note is the large quantity of packages that require installation, and the users complained about the time spent during such installation, but praised the closed environment that it was configured into, saying that although many packages were installed, they would not affect anything outside the environment of the system, and consequently other projects they might have, thus proving once again the modularity of MoQA.

As for a final assessment, the system may not run as specified as it should on Windows systems, since the Genia Tagger software is not compatible with that operating system. Still, MoQA uses Stanford NER as its default NER, so the behavior of the system remains unaltered.

4.4.3.5 Summary

These results show that as time progressed, so did the systems developed within the MoQA framework. MoQABio represents a considerable improvement when compared to the first one, with improvements of a minimum of 5 times MoQA and 25 times their ancestor, WS4A. Although it is not possible to perform a direct comparison between the results presented and the ones in from the organization due to lack of official results for the general competition, the results show much promise, with the features, parser and tools selected as the main reason for the performance improvement, since they all merged together, from the parsing to the document search, to the DBpedia and OpenPHACTS requests, to the classifier.

Chapter 5

Conclusion

This work's goal was achieved by creating an initial solution to some of the problems existing in QA systems today. This solution was achieved by creating a framework embedded with long researched tools and a modular architecture that enables the proven sub-modules to intertwine among themselves to achieve user specific goals. The modularity of such system was achieved by maintaining a strict policy between modules, with clear input and output, and a configuration file which is user made. The most difficult tasks were maintaining availability across different systems, use tools that have a short learning curve, and keep the prerequisites to a minimum, or at least make them easily configurable.

The developed work enables users to ask questions and obtain their correspondent answer, along with their documents, given that an annotated file, parser and mapping configurations are provided. Also, in order to conjugate the modules, it must have a scripting file that includes and glues together every module, their input and output, registry files and classification files.

The classification module of the developed work is based on supervised machine learning algorithms, namely SVM, which was the preferred and most successful choice, along with specified and generalized parsers such as the GENIA tagger and Stanford NER. These models are generated using several features from a wide selection developed over a rather hefty quantity of research of the area and of previous works among the competitions in which the system took part. Through them, answer classification modules were developed, distinguishing useful from no useful articles, given the question and asnwer annotations.

A biomedical use case for a QA system was developed, MoQABio, that showed improved results over MoQA, the non specified one. MoQABio had specific annotators and features in its classification module that enabled improved results even against systems that had all access to PubMed documents, compared to MoQABio that had a much restricted access, due to its objective of being as much local as possible. A simple web application with a prototypical user interface was also developed with a field for inputting

a question and getting the results much like YodaQA.

The final system, MoQABio, is the result of two iterations that were used to participate in the BioASQ 2016 and SemEval of 2017 international workshops. These participation allowed the assessment of the system officially, comparing it to the systems submitted by other teams across the globe. The local results of MoQABio represent a major improvement when compared to the close to bottom results achieved previously with WS4A, for the identification of correct documents and snippets.

Ultimately, three systems were compared using the same datasets for training and testing. The second system in this ultimate comparison resulted in a performance increase of more than ten-fold when compared to the first one. This discrepancy is a result of a better choice of training features and parser, which resulted in an increase (in percentage points) of 1617 in precision, 2486 in recall and 1920 in f-measure in the document scores and 1800 in precision, 1400 in recall and 2300 in f-measure in the snippet scores. The third and last installment consisted on a fine tuning of the system leading to another increase of percentage in precision, recall, f-measure, comparing to the first system of 2550 points in precision, 4429 points in recall and 3729 points in f-measure in the document scores. As for snippets, the percentage point increase consisted of 4150 points in precision, 7100 points in recall and 4650 points in f-measure. Comparing the second system to the last one the percentage point increase was of 54 in precision, 75 in recall and 89 in f-measure in the document scores and 124 in precision, 380 in recall and 98 in f-measure in the snippet scores.

MoQA's focus was on modularity, but still, the answers retrieved are to be minimally accepted or at least give some guidelines of what the answers should be. The system, therefore, works as a sort of mixture of black box and white box, where one may use MoQA for some already defined purposes, such as MoQABio and MoRS, without knowing the works of the system, while another kind of future users may edit MoQA in order to acquiesce to their needs.

5.1 Future Work

As the results also shown, there is still a lot of room for improvement for QA systems. Some of those improvements include the handling of empty questions, the better use of metadata to better grab a context and direction of the question in itself. This would also result in a better recognition of correct triples and concepts. The system developed is to be modular to any domain knowledge besides the biomedical example, given the correct dataset is provided. It also has the ability to adapt to general questions without a specific domain knowledge, thanks to the implementation of DBpedia [1] RDF queries through a Web Service.

[1] http://dbpedia.org

New features (i.e., part-of-the-speech, lemmas.) can also be explored to enhance the set of features used during the training, thus improving the correct document identification. The pipeline architecture designed for question answering has proven modular, however, some modules need to be more robust, in order to better handle exceptions and ease even further the development using MoQA. This could be solved by having secondary tools ready to be used, in case the first option was not available or delivered a deficient result. In addition, better logical and clearer module separation is also suggested, to ease development.

As always, time goes by and tools and technologies get outdated, or are replaced with better ones. The Genia tagger, in spite of excellent results with the Genia corpus, is almost eleven years old, and a new solution in this area may be necessary, and training a Stanford NER instance with the Genia corpus and other biomedical corpora are the next step in this direction.

The incorporation of other Sematch measures, with attention as to the time spent on them, so that the elapsed time doesn't dramatically increase.

Further experiments, such as feature ablation, might also give better insight of what kind of features make the most difference in scoring measurements.

As for the searching of documents within Elasticsearch, a suggestion for improving the search of documents (more specifically articles) would be further improving the search by adding a learning to rank algorithm, using the work present at `https://github.com/o19s/elasticsearch-learning-to-rank`, and building a model tailored to the developers wishes.

Due to the amount of biomedical information available, namely biomedical articles and controlled vocabularies, this work leverage entirely on English native language data sources. Future work must be done to evaluate if this system is able to recognize and normalize biomedical entities within other languages, namely in Portuguese articles. This presents a rather ambitious challenge as fewer sources of Portuguese biomedical data are available, and the controlled vocabularies are scarce. Some of the tools used in MoQA already support the Portuguese language, such as NLTK and Sematch, which support multiple idioms.

Finally, the user interface could be upgraded, by showing more graphical similarities between the concepts, triples, and showing graphic images of the articles and even the concepts that make up the answer provided. More details about the metadata from external sources would certainly enrich the answer, and also the suggestion of similar questions so one may further one's research of a specific topic by making at least one follow-up question.

Bibliography

[1] Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D. Lawrence, David C. Gondek, and James Fan. Learning to rank for robust question answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 833–842, New York, NY, USA, 2012. ACM.

[2] Gabor Angeli, Neha Nayak, and Christopher D. Manning. Combining natural logic and shallow reasoning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 442–452, Berlin, Germany, August 2016. Association for Computational Linguistics.

[3] Yuichiro Anzai. *Pattern recognition and machine learning*. Elsevier, 2012.

[4] Chidanand Apte, Fred Damerau, Sholom M Weiss, Chid Apte, Fred Damerau, and Sholom Weiss. Text mining with decision trees and decision rules. In *In Proceedings of the Conference on Automated Learning and Discorery, Workshop 6: Learning from Text and the Web*. Citeseer, 1998.

[5] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[7] Jason Baldridge. The opennlp project. *URL: http://opennlp. apache. org/index. html,(accessed 2 February 2012)*, 2005.

[8] Georgios Balikas, Aris Kosmopoulos, Anastasia Krithara, Georgios Paliouras, and Ioannis Kakadiaris. Results of the bioasq tasks of the question answering lab at clef 2015. In *CLEF 2015*, 2015.

[9] Daniel Bär, Torsten Zesch, and Iryna Gurevych. Dkpro similarity: An open source framework for text similarity. In *ACL (Conference System Demonstrations)*, pages 121–126, 2013.

[10] Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015.

[11] Matthew W Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 459–468. ACM, 2010.

[12] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[14] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[15] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010.

[16] Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. Qakis: an open domain qa system based on relational patterns. In *Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914*, pages 9–12. CEUR-WS. org, 2012.

[17] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[18] Sungbin Choi. Snumedinfo at clef qa track bioasq 2015. In *CLEF 2015*, 2015.

[19] Peter Christen. A comparison of personal name matching: Techniques and practical issues. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 290–294. IEEE, 2006.

[20] Aaron M Cohen and William R Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, 2005.

[21] UniProt Consortium et al. Uniprot: a hub for protein information. *Nucleic acids research*, page gku989, 2014.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[23] Francisco M Couto and H Sofia Pinto. The next generation of similarity measures that fully explore the semantics in biomedical ontologies. *Journal of bioinformatics and computational biology*, 11(05):1371001, 2013.

[24] Francisco M Couto and Mário J Silva. Disjunctive shared information between ontology concepts: application to gene ontology. *Journal of biomedical semantics*, 2(1):5, 2011.

[25] G. Da San Martino, A. Barrón-Cedeño, S. Romeo, A. Moschitti, S. Joty, F. A. A. Obaidli, K. Tymoshenko, and A. Uva. Addressing Community Question Answering in English and Arabic. *ArXiv e-prints*, October 2016.

[26] Andreas Doms and Michael Schroeder. Gopubmed: exploring pubmed with the gene ontology. *Nucleic acids research*, 33(suppl 2):W783–W786, 2005.

[27] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[28] Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618. Citeseer, 2013.

[29] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[30] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[31] Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. In *SemEval@NAACL-HLT*, 2016.

[32] E Garcia. Cosine similarity and term weight tutorial. *Information retrieval intelligence*, 2006.

[33] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[34] Martin Gleize and Brigitte Grau. Limsi-cnrs@clef 2015: Tree edit beam search for multiple choice question answering. In *CLEF 2015*, 2015.

[35] D. Goncalves, M. Costa, and F. M. Couto. A Flexible Recommendation System for Cable TV. *ArXiv e-prints*, September 2016.

[36] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.", 2015.

[37] Vishal Gupta, Gurpreet S Lehal, et al. A survey of text mining techniques and applications. *Journal of emerging technologies in web intelligence*, 1(1):60–76, 2009.

[38] Sherzod Hakimov, Hakan Tunc, Marlen Akimaliev, and Erdogan Dogdu. Semantic question answering system over linked data using relational patterns. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 83–88. ACM, 2013.

[39] Felix Hieber and Stefan Riezler. Improved answer ranking in social question-answering portals. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, SMUC '11, pages 19–26, New York, NY, USA, 2011. ACM.

[40] Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of question answering in the semantic web. *Submitted to the Semantic Web Journal*, 2016.

[41] Chang'e Jia, Xinkai Du, Chengjie Sun, and Lei Lin. Itnlp-aikf at semeval-2016 task 3: a question answering system using community qa repository. *Proceedings of SemEval*, pages 904–909, 2016.

[42] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[43] Thorsten Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[44] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[45] Vlado Keselj. Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6. *Computational Linguistics*, 35(3):463–466, 2009.

[46] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics, 2004.

[47] Grzegorz Kondrak. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer, 2005.

[48] Anastasia Krithara, Anastasios Nentidis, George Paliouras, and Ioannis Kakadiaris. Results of the 4th edition of bioasq challenge. In *Proceedings of the Fourth BioASQ workshop at the Conference of the Association for Computational Linguistics*, pages 1–7, 2016.

[49] John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

[50] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360, 2016.

[51] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[52] Robert Leaman, Graciela Gonzalez, et al. Banner: an executable survey of advances in biomedical named entity recognition. In *Pacific symposium on biocomputing*, volume 13, pages 652–663, 2008.

[53] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[54] Yuhua Li, Zuhair A Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on knowledge and data engineering*, 15(4):871–882, 2003.

[55] Dekang Lin et al. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304, 1998.

[56] Gareth F. Jones Linda Cappellato, Nicola Ferro and Eric San Juan. Preface. In *CLEF 2015*, 2015.

[57] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[58] Haibin Liu, Tom Christiansen, William A Baumgartner, and Karin Verspoor. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *Journal of biomedical semantics*, 3(1):3, 2012.

[59] MultiMedia LLC. MS Windows NT kernel description, 1999.

[60] Henry J Lowe and G Octo Barnett. Understanding and using the medical subject headings (mesh) vocabulary to perform literature searches. *Jama*, 271(14):1103–1108, 1994.

[61] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[62] Edgard Marx, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Konrad Höffner, Jens Lehmann, and Sören Auer. Towards an open question answering architecture. In *Proceedings of the 10th International Conference on Semantic Systems*, pages 57–60. ACM, 2014.

[63] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA, 2010.

[64] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[65] Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, et al. Super team at semeval-2016 task 3: Building a feature-rich system for community question answering. *Proceedings of SemEval*, pages 836–843, 2016.

[66] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[67] David Milne and Ian H Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[68] Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, D. Scott Cyphers, and Jim Glass. Sls at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *SemEval@NAACL-HLT*, 2016.

[69] Preslav Nakov, Doris Hoogeveen, Lluís Márquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[70] Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545, San Diego, California, June 2016. Association for Computational Linguistics.

[71] Preslav Nakov, Lluís Márquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June 2016. Association for Computational Linguistics.

[72] Mariana Neves. Hpi question answering system in the bioasq 2015 challenge. In *CLEF 2015*, 2015.

[73] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, page gkp440, 2009.

[74] Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, pages 82–86. Morgan Kaufmann Publishers Inc., 2002.

[75] Arzucan Ozgür. *Supervised and unsupervised machine learning techniques for text document categorization*. PhD thesis, Citeseer, 2004.

[76] Sujit Pal. Computing semantic similarity for short sentences. 2014.

[77] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[78] Jan Sedivý Petr Baudis. Biomedical question answering using the yodaqa system: Prototype notes. In *CLEF 2015*, 2015.

[79] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on systems, man, and cybernetics*, 19(1):17–30, 1989.

[80] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[81] Björn Rudzewitz Ramon Ziai. Comic: Exploring text segmentation and similarity in the english entrance exams task. In *CLEF 2015*, 2015.

[82] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[83] Dietrich Rebholz-Schuhmann, Miguel Arregui, Sylvain Gaudan, Harald Kirsch, and Antonio Jimeno. Text processing through web services: calling whatizit. *Bioinformatics*, 24(2):296–298, 2008.

[84] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[85] Miguel J. Rodrigues and Francisco M Couto. Mors at semeval-2017 task 3: Easy to use svm in ranking tasks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 278–282, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[86] Miguel J. Rodrigues, Miguel Falé, Andre Lamurias, and Francisco M. Couto. WS4A: a biomedical question and answering system based on public web services and ontologies. *CoRR*, abs/1609.08492, 2016.

[87] Denis Savenkov. Ranking answers and web passages for non-factoid question answering: Emory university at trec liveqa.

[88] Eric Sayers and David Wheeler. Building customized data pipelines using the entrez programming utilities (eutils). 2004.

[89] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM, 2015.

[90] Fact Sheet. Medline. *National Library of Medicine (US)*, 2004.

[91] Apache Solr. Apache solr, 2011.

[92] RL Somorjai, M Alexander, R Baumgartner, S Booth, C Bowman, A Demko, B Dolenko, M Mandelzweig, AE Nikulin, N Pizzi, et al. Artificial intelligence methods and tools for systems biology, 2004.

[93] Rob Stewart. A demonstration of a natural language query interface to an event-based semantic web triplestore. *The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*, 8798:343, 2014.

[94] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[95] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. ACM, 2015.

[96] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37:351–383, 2011.

[97] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda. Svm answer selection for open-domain question answering. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[98] Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics, 2000.

[99] Quan Hung Tran, Vu Tran, Tu Vu, Minh Le Nguyen, and Son Bao Pham. Jaist: Combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*, volume 15, pages 215–219, 2015.

[100] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138, 2015.

[101] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1, 2015.

[102] Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. Developing a robust part-of-speech tagger for biomedical text. In *Panhellenic Conference on Informatics*, pages 382–392. Springer, 2005.

[103] Ricardo Usbeck, Erik Körner, and Axel-Cyrille Ngonga Ngomo. Answering boolean hybrid questions with hawk.

[104] Menno van Zaanen. Multi-lingual question answering using openephyra. In *CLEF (Working Notes)*, 2008.

[105] Di Wang and Eric Nyberg. CMU OAQA at TREC 2015 liveqa: Discovering the right answer with clues. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*, 2015.

[106] Ke Wang, Ning Liu, Iman Sadooghi, Xi Yang, Xiaobing Zhou, Tonglin Li, Michael Lang, Xian-He Sun, and Ioan Raicu. Overcoming hadoop scaling limitations through distributed task execution. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 236–245. IEEE, 2015.

[107] Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, and Stephen H Bryant. Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, page gkp456, 2009.

[108] Jonathan J Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on Computational linguistics-Volume 4*, pages 1106–1110. Association for Computational Linguistics, 1992.

[109] Chih-Hsuan Wei, Robert Leaman, and Zhiyong Lu. Beyond accuracy: creating interoperable and scalable text-mining web services. *Bioinformatics*, page btv760, 2016.

[110] Antony J Williams, Lee Harland, Paul Groth, Stephen Pettifer, Christine Chichester, Egon L Willighagen, Chris T Evelo, Niklas Blomberg, Gerhard Ecker, Carole Goble, et al. Open phacts: semantic interoperability for drug discovery. *Drug discovery today*, 17(21):1188–1198, 2012.

[111] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[112] Pak Chung Wong, Paul Whitney, and Jim Thomas. Visualizing association rules for text mining. In *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 120–123. IEEE, 1999.

[113] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[114] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[115] Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. Answering natural language questions via phrasal semantic parsing. In *Natural Language Processing and Chinese Computing*, pages 333–344. Springer, 2014.

[116] Zi Yang, Niloy Gupta, Xiangyu Sun, Di Xu, Chi Zhang, and Eric Nyberg. Learning to answer biomedical factoid & list questions: Oaqa at bioasq 3b. In Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors, *CLEF (Working Notes)*, volume 1391 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[117] Zi Yang, Niloy Gupta, Xiangyu Sun, Di Xu, Chi Zhang, and Eric Nyberg. Learning to answer biomedical factoid and list questions oaqa at bioasq 3b. In *Working Notes for the Conference and Labs of the Evaluation Forum (CLEF), Toulouse, France*, 2015.

[118] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.

[119] Ganggao Zhu and Carlos A Iglesias. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, 2017.

[120] Ganggao Zhu and Carlos Angel Iglesias Fernandez. Sematch: semantic entity search from knowledge graph. 2015.